(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) Patentschrift

(10) DE 39 26 327 C 2

(51) Int. Cl.[6]:
G 06 K 9/60
G 06 K 9/34

(21) Aktenzeichen: P 39 26 327.4-53
(22) Anmeldetag: 9. 8. 89
(43) Offenlegungstag: 15. 2. 90
(45) Veröffentlichungstag der Patenterteilung: 18. 3. 99

DE 39 26 327 C2

Innerhalb von 3 Monaten nach Veröffentlichung der Erteilung kann Einspruch erhoben werden

(30) Unionspriorität:
230847 10. 08. 88 US

(73) Patentinhaber:
Caere Corp., Los Gatos, Calif., US

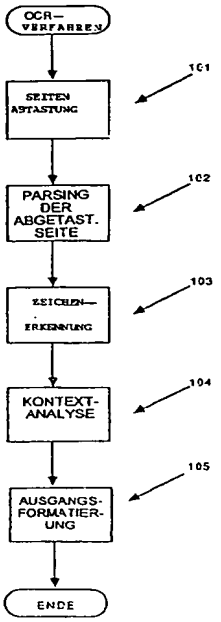
(74) Vertreter:
Zenz, Helber, Hosbach & Partner, 45128 Essen

(72) Erfinder:
Bernzott, Philip, Oakland, Calif., US; Dilworth, John, Oakland, Calif., US; George, David, Oakland, Calif., US; Higgins, Bryan, Berkeley, Calif., US; Knight, Jeremy, Berkeley, Calif., US

(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:
BAYER, T., BLÄSIUS, K.-H.: "Regelgesteuerte Zeichenerkennung und Dokumentklassifikation" Mustererkennung 1987, 9. DAGM-Symposium Braunschweig, Springer-Verlag, S. 88;
WONG, K.Y., CASEY, R.G., WAHL, F.M.: "Document Analysis System" IBM J.Res.Develop., Vol. 26, No. 6, Nov. 1987, S. 647-656;
ZIMMERMANN, R.: "Automatische Erkennung von Schriftzeichen" VDI-Z 115 (1973) Nr. 11, August, S. 875-885;

(54) Verfahren und System zum Erkennen von Zeichen auf einem Medium

(57) Verfahren zum Erkennen von Zeichen auf einem Medium
wobei das Medium abgetastet und ein Bit-mapped-Bild des Mediums erzeugt wird;
wobei das Bit-mapped-Bild einer Strukturanalyse (103, 203, 232) zum Isolieren einzelner Zeichen unterworfen wird und als Ausgabe der Strukturanalyse ein Bit-mapped-Bild der Zeichen erzeugt wird;
wobei die Zeichen vorläufig identifiziert werden (234); und
wobei jedes Zeichen auf der Basis des das Zeichen im Medium umgebenden Kontextes analysiert wird (104);
dadurch gekennzeichnet,
a) daß eine Textzeile in dem Medium zum Bestimmen von räumlichen Informationen über die Textzeile analysiert wird;
b) daß Attributdaten für jedes vorläufig identifizierte Zeichen in der Textzeile gewonnen und diesem Zeichen zugeordnet werden; und
c) daß Mehrdeutigkeiten bei einem vorläufig identifizierten Zeichen mit Hilfe der räumlichen Informationen über die Textzeile und mit Hilfe der zugeordneten Attributdaten aufgelöst werden (1502).



DE 39 26 327 C2

Beschreibung

Die Erfindung bezieht sich auf das Gebiet der optischen Zeichen-Erkennungssysteme und insbesondere auf ein Verfahren und eine Anordnung zum Erkennen von Zeichen auf einem Medium nach den Oberbegriffen der Ansprüche 1 und 5.

Eine Anzahl optischer Zeichen-Erkennungssysteme (OCR-Systeme) sind bekannt. In typischer Ausführung weisen derartige Systeme einen Scanner zum Abtasten einer Seite eines gedruckten Textes und zur Durchführung einer Zeichenerkennung an einem Bit-mapped Bild des Textes auf (vgl. R. ZIMMERMANN: "Automatische Erkennung von Schriftzeichen", VDI-Z 115 (1973) Nr. 11, August, Seiten 875–885). Die Zeichen können danach in einer Datei eines Computersystems zur Weiterverarbeitung durch ein Textverarbeitungsprogramm oder ähnliches gespeichert werden.

Einige bekannte OCR-Systeme enthalten einen Handscanner zum Abtasten einer Seite. Bei derartigen Systemen streicht die Bedienungsperson bei der Durchführung der Abtastung mit dem Handgerät über den gedruckten Text auf der Seite, wobei normalerweise das Abtasten graphischer oder nicht-Text-enthaltender Abschnitte auf der Seite vermieden wird. Normalerweise wird die Seite in der Weise abgetastet, in der sie normalerweise gelesen wird (d. h. das Scannen erfolgt entlang der Spalten und spaltenweise von links nach rechts).

Andere bekannte Systeme enthalten eine Entscheidungseinrichtung, die zum Feststellen und Angeben derjenigen Teile des Textes benutzt werden kann, welche von dem OCR-System zu verarbeiten sind. Einige dieser Systeme sind in der Lage, graphische Abschnitte der angezeigten Seitenzonen von Textabschnitten zu unterscheiden. Derartige Systeme bedingen jedoch dennoch einen manuellen Eingriff zum Markieren des Textes in der Richtung oder Reihenfolge des normalen Lesens und zum Markieren von Graphikabschnitten.

Andere Systeme machen von einer Registrierungsmarke zum Anzeigen des Beginns von Textspalten Gebrauch. Diese Systeme erfordern ebenfalls noch manuelle Eingriffe zum Setzen von Registrierungsmarken.

K. Y. Wong, R. G. CASEY und F. M. WAHL beschreiben in dem Aufsatz "Document Analysis System", IBM J. Res.Develop., Vol. 26, No. 6, Nov. 1982, Seiten 647–656, ein OCR System, das in der Lage ist, eine Seite mit gemischten Text- und nicht-Text-Informationen abzutasten und zu erkennen, sowie zwischen Text- und nicht-Text-Bereichen zu unterscheiden.

Bekannte optische Zeichen-Erkennungssysteme können in zwei Kategorieren eingeteilt werden. Optische Zeichen-Erkennungssysteme der ersten Kategorie erkennen entweder eine einzelne Schriftart oder eine begrenzte Anzahl von Schriftarten, und ihre Eingabe ist gewöhnlich beschränkt auf eine Schriftart mit konstantem Zeichenabstand bei einer speziellen Punktgröße. Optische Zeichen-Erkennungssysteme der zweiten Kategorie werden typischerweise als Allschriftsysteme (omnifont systems) bezeichnet. Derartige Systeme sind in der Lage, eine große Anzahl von Schriftarten in einem großen Bereich von Punktgrößen entweder konstanten oder proportionalen Abstandes zu erkennen. Generell sind optische Zeichen-Erkennungssysteme, die eine große Anzahl von Schriften erkennen können, nicht in der Lage, Dokumente mit hohen Geschwindigkeiten zu verarbeiten, wie sie beispielsweise bei Systemen zur Erkennung einer begrenzten Anzahl spezieller Schriften realisierbar sind.

Der Erfindung liegt die Aufgabe zugrunde, ein Verfahren und eine Anordnung zum Erkennen von Zeichen auf einem Medium mit erhöhter Genauigkeit bei der Zeichenerkennung zu schaffen.

Diese Aufgabe wird erfindungsgemäß durch ein Verfahren mit den Merkmalen des Anspruchs 1 sowie mit der im Anspruch 5 angegebenen Anordnung gelöst. Vorteilhafte Weiterbildungen der Erfindung sind in den Unteransprüchen gekennzeichnet.

Die Erfindung vereinigt die bisher als gegensätzlich angesehenen Eigenschaften der Erkennung irgendeiner Anzahl von Schriftarten und der raschen Dokumentenverarbeitung.

Das erfindungsgemäße optische Zeichen-Erkennungssystem weist eine Scannereinrichtung zum Abtasten eines Dokumentes und zum Erzeugen eines Bit-mapped Bildes des Dokuments auf. Die Scannereinrichtung ist mit einem Computersystem gekoppelt, das einen Speicher zur Speicherung des Bit-mapped Bildes und einen Prozessor zum Verarbeiten des Bit-mapped Bildes und zur Ausgabe von Daten aufweist, die die Zeichen auf der abgetasteten Seite darstellen.

Die Erfindung wird vorteilhaft mit Verfahrensschritten verwendet, die das Abtasten der Seite und die Erkennung und Ausgabe von Zeichen auf der Seite in einer Reihenfolge ermöglicht, welche logisch die gleiche Reihenfolge ist, die eine die Seite lesende Person verfolgen würde. Dies geschieht durch Parsing (automatische Zergliederung mittels Strukturanalyse) der Seite in eine Vielzahl von Blöcken und Ausgabe der Blöcke an einen Zeichen-Erkennungsprozeß in einer Reihenfolge, die gewöhnlich die logische Lesefolge auf der Seite ist.

Sinnvollerweise wird ein Zeichen-Erkennungsverfahren aus einer Kombination eines Schablonen-Anpaßprozesses und eines Merkmalsanalyseprozesses eingesetzt. Der Merkmalsanalyseprozeß ermöglicht die Zeichenerkennung auf der Basis ihrer Formen. Unter Verwendung des Merkmalsanalyseprozesses ist es möglich, Zeichen in einer beliebigen Anzahl unterschiedlicher Schriften zu erkennen. Durch Verwendung des Schablonen-Anpaßprozesses ist in Verbindung mit dem Merkmalsanalyseprozeß ein vernünftiger Dokumentendurchsatz möglich, und zwar ohne die Notwendigkeit einer Schablonenbibliothek.

T. BAYER, K.-H. BLÄSIUS beschreiben in ihrem Aufsatz "Regelgesteuerte Zeichenerkennung und Dokumentklassifikation" Mustererkennung 1987, 9. DAGM-Symposium Braunschweig, Springer-Verlag, Seite 88, eine Kontextanalyse, die den Kontext von Einzelzeichen im Text analysiert, um Fehler in der Einzelzeichenklassifikation zu beheben. Ansatzpunkte sind hierbei geometrische Zusammenhänge in Layoutstrukturen, die etwa eine Korrektur von Groß- in Kleinbuchstaben erlauben oder unsichere Klassifikationsergebnisse zu sicheren Entscheidungen umformen. Mit N-Grammen und Wörterbüchern lassen sich die Textinhalte auf orthographische Fehler oder Segmentierfehler untersuchen und verbessern.

Erfindungsgemäß wird von einem Kontext-Analyseprozeß Gebrauch gemacht, der räumliche Informationen über die Textzeilen sowie genuine Attribute der Schriftzeichen verwendet.

Das Zeichen-Erkennungsverfahren wird durch eine iterative Auflösung von Mehrdeutigkeiten in der Form vervollständigt, wodurch die Zahl der typographischen oder semantischen (Schreib- oder Bedeutungs-)Fehler minimiert wird.

Die restlichen Unstimmigkeiten [redacted]en als Identifizierungen auf geringem Zuverläss[redacted]tsniveau gekennzeichnet.

Angegeben werden außerdem einige Techniken zur Erzeugung einer Schablonenübereinstimmung, z. B. die Erzeugung von Darstellungen von Zeichen mit Bits in einem Bit-mapped Bild des Zeichens, die für das zu erkennende Zeichen dunkel- oder ausgeschaltet sein müssen, sowie zur Erzeugung von Darstellungen mit Bits, die ein- oder hellgeschaltet sein müssen. Diese Technik ermöglicht die Erkennung von Zeichen innerhalb gewisser Toleranzen. Außerdem wird ein Verfahren zur Zeichenerkennung unter Verwendung von Zeichenschablonen (character templates) für den Fall beschrieben, daß Zeichen eng zusammengerückt sind.

Außerdem wird die Verwendung mehrerer Routinen bei dem Merkmalsanalyseprozeß beschrieben. Jede dieser Routinen dient der Erkennung einer Zeichenform. Angegeben werden Verfahren zur Beschreibung von Zeichen mit statistischer Information und zur Anpassung von Polygonen an verschiedene Erscheinungsformen des Zeichens. Auf der Basis dieser statistischen Information und der Polygone können Merkmalsanalyseprozesse die Formen der Zeichen erkennen.

Beschrieben werden außerdem Verfahren zur Unterscheidung von Text und Graphik auf einem Dokument. Während einer Dokumentenverarbeitung zum Zwecke der Zeichenerkennung werden Graphik enthaltende Zonen eines Dokuments identifiziert, die während des Erkennungsprozesses außer Betracht bleiben.

Angegeben werden auch Möglichkeiten zur Messung der relativen Beschaffenheit, Struktur bzw. Textur kleiner Zonen des Dokuments, um festzustellen, ob die Zone Text oder Graphik enthält.

Im folgenden wird die Erfindung anhand von in der Zeichnung schematisch dargestellten Ausführungsbeispielen näher erläutert. In der Zeichnung zeigen:

Fig. 1 ein Gesamtablaufdiagramm des optischen Zeichen-Erkennungsverfahrens;

Fig. 2(a) ein Ablaufdiagramm eines Gesamt-Parsingverfahrens bei dem optischen Zeichen-Erkennungsverfahren gemäß **Fig.** 1;

Fig. 2(b) ein Ablaufdiagramm eines Seiten-Parsingverfahrens;

Fig. 2(c) ein Ablaufdiagramm eines Block-Parsingverfahrens;

Fig. 2(d) ein Ablaufdiagramm eines Zeilen-Parsingverfahrens;

Fig. 3(a) einen Teil einer Seite, wie sie mit dem beschriebenen Zeichen-Erkennungsverfahren verarbeitet werden kann;

Fig. 3(b) ein Bit-mapped Bild des Teils der Seite gemäß **Fig.** 3(a);

Fig. 4 ein Ablaufdiagramm eines Verfahrens zur Erzeugung eines Text-Map-Feldes;

Fig. 5 ein Ablaufdiagramm eines Verfahrens zur Entzerrung einer Seite;

Fig. 6 ein Ablaufdiagramm eines Verfahrens zur Lokalisierung von Wegen auf einer abgetasteten Seite;

Fig. 7 ein Blockdiagramm, das zwei Datenstrukturen darstellt;

Fig. 8 ein Ablaufdiagramm eines Verfahrens zur Lokalisierung von Blöcken;

Fig. 9 eine erste Seitenabbildung, unterteilt in mehrere Blöcke;

Fig. 10(a) eine zweite Seitenabbildung, unterteilt in mehrere Blöcke;

Fig. 10(b) eine zweite Darstellung der zweiten Seitenabbildung, wie sie abgetastet werden kann;

Fig. 11(a) ein Gesamtablaufdiagramm eines Zeichen-Erkennungsverfahrens, das bei der Erfindung verwendbar ist;

Fig. 11(b) ein Ablaufdiagramm eines Schablonenanpaßprozesses;

Fig. 12(a) eine Darstellung eines Bit-mapped Bildes eines Zeichens;

Fig. 12(b) eine Darstellung einer zweiten Version des Bit-mapped Bildes eines Zeichens;

Fig. 13 ein Ablaufdiagramm eines Merkmalsanalyseprozesses;

Fig. 14(a) eine Darstellung eines ersten Zeichenfensters;

Fig. 14(b) eine Darstellung eines zweiten Zeichenfensters;

Fig. 14(c) ein Polygonanpaßverfahren; und

Fig. 15 ein Ablaufdiagramm eines Kontext-Analyseprozesses.

In der folgenden Beschreibung werden zahlreiche Einzelheiten, wie Pixeldichten, Bytegrößen usw. angegeben, um das Verständnis für die Erfindung zu vertiefen. Es ist jedoch für den Fachmann klar, daß die Erfindung auch ohne diese besonderen Einzelheiten realisiert werden kann. In anderen Fällen werden bekannte Schaltungen, Strukturen und Techniken nicht im einzelnen beschrieben, um die Erfindung nicht mit unnötigen Erläuterungen zu belasten.

Im folgenden wird auf **Fig.** 1 Bezug genommen. Dort ist ein Gesamtablaufdiagramm des erfindungsgemäßen Verfahrens gezeigt. Erfindungsgemäß wird zunächst eine Seite abgetastet, Block **101**. Wie beschrieben werden wird, ist die Erfindung zum Trennen und Übersetzen des Textes auf einer Seite nicht nur dann in der Lage, wenn die Seite ausschließlich Text enthält, sondern auch dann, wenn die Seite eine Kombination von Text- und sonstigen Zonen enthält. Außerdem braucht die Erfindung keine manuellen Eingriffe zur Anzeige der normalen Reihenfolge des Lesens des Textes.

Nach dem Abtasten der Seite wird die Seite geparst (syntaxanalisiert), Block **102**. Das Parsing der Seite wird weiter unten genauer beschrieben und kann generell in die Funktionen des Seiten-Parsing, des Block-Parsing und des Zeilen-Parsing unterteilt werden. Nach dem Parsing der Seite werden die Formen der einzelnen Zeichen mit Hilfe eines Zeichen-Erkennungsprozesses, Block **103**, festgestellt. Die Erfindung ist in der Lage, Zeichen einer beliebigen Anzahl von Schriftarten zu erkennen.

Ein als Kontextanalyse bezeichneter Vorgang dient zur Prüfung der relativen Größen und Positionen der während des Zeichen-Erkennungsprozesses festgestellten Formen, um den Text in Worte zu unterteilen und Unsicherheiten der Form zu klären, Block **104**.

Schließlich werden die erkannten Zeichen für die Ausgabe formatiert, Block **105**.

SCANNER

Das beschriebene Verfahren und die Einrichtung sind in der bevorzugten Ausführung so konzipiert, daß sie in Verbindung mit im Handel erhältlichen Mikroprozessoren mit 32-Bit-Adreßräumen arbeiten. Beispiele für solche Mikroprozesoren sind die unter den Warenzeichen Motorola 68020 und Intel 80386 erhältliche Mikroprozessoren.

Es ist für den Fachmann klar, daß die Erfindung mit einer beliebigen Anzahl verschiedenartiger unterschiedlicher Computersysteme mit einem Prozessor und einem Speicher realisiert werden kann.

Als Scanner kann einer der verschiedenen bekannten Scanner oder ein zukünftig noch zu entwickelnder Scanner eingesetzt werden. Die Erfindung ist konzipiert dafür, daß sie in Verbindung mit einem Scanner arbeitet, der eine Seite gedruckter Information abzutasten vermag und ein Bit-mapped Bild der Seite erzeugt. Vorzugsweise können bei der Erfindung preisgünstige optische Scanner in Verbindung mit ebenfalls preisgünstigen Personal Computersystemen verwendet werden, um das optische Zeichen-Erkennungssystem kostengünstig und wirtschaftlich zu machen.

PARSING

Nach der Abtastung einer Seite wird ein Bit-mapped Bild der Seite im Speicher eines Computersystems oder in anderen zur Speicherung von Bit-mapped Bildern geeigneten Systemen gespeichert. Gemäß **Fig.** 2(a) beginnt der Parsing-Prozeß mit dem Seitenparsing, Block **201**. Der Seiten-Parsingvorgang arbeitet an dem Bit-mapped Bild und teilt die Seite in eine Vielzahl von Blöcken. Beim Seiten-Parsingprozeß wird versucht, die nicht-leeren Teile der Seite in eine Vielzahl von Blöcken zu unterteilen und zwischen Text und nicht-Text oder Graphik zu unterscheiden. Mit dem Seiten-Parsing wird versucht, sicherzustellen, daß ein einzelner Block entweder nur Text oder nur nicht-Text enthält. Nicht-Text-Blöcke werden von der Weiterverarbeitung ausgeschlossen.

Bei dem Seiten-Parsingprozeß, Block **201**, wird die Struktur oder Textur des abgetasteten Bildes zum Trennen von Text und Graphik und zur Feststellung des Vorhandenseins von Spalten und Kopfzeilen analysiert. Auf der Basis eines normalisierten, zweidimensionalen Maßstabs der örtlichen Dichte von schwarz-weiß-Übergängen werden Bereiche auf der Seite entweder als Graphik, als Text oder als Linien charakterisiert. Die Bereiche werden dadurch in Blöcke gruppiert, daß Wege von weißen Räumen gefunden werden, die Bereiche ähnlicher Textur oder Beschaffenheit (z. B. mit ähnlicher Dichte von schwarz-weiß-Übergängen) umgeben. Wenn sie nicht durch Linien oder übergroße weiße Räume getrennt sind, werden benachbarte Blöcke mit ähnlichen Beschaffenheits- und Strukturcharakteristiken und Ausrichtung zusammengefaßt. Der Seiten-Parsingprozeß, Block **201**, analysiert danach die relative Lage von Blöcken, um festzustellen, welche Blöcke andere Blöcke nach Zeilenhöhe, Blockbreite und Vertikalposition auf der Seite überlappen. Der Seiten-Parsingprozeß kann danach eine vernünftige Interpretation des Seitenlayout konstruieren. Die endgültige Ausgabe des Seiten-Parsingprozeßblocks **201** ist ein geordneter Satz von Blockdeskriptoren, welche die normale Lesefolge nachzeichnen, die durch das ermittelte Seitenlayout impliziert wird.

Ein Block-Parsingprozeßblock **202** analysiert die schwarz-weiß-Übergänge zur Berechnung des Grades des Versatzes (skew) und zur Lokalisierung horizontaler Wege, welche einen vorgegebenen Textblock in einzelne Textzeilen unterteilen. Der Block-Parsingprozeß, Block **202**, bestimmt Vertikallinien und eliminiert die Linien von der weiteren Verarbeitung.

Der Zeilen-Parsingprozeß, Block **203**, sucht jede Zeile von links nach rechts ab und lokalisiert vertikale Freiräume. Derartige vertikale Freiräume trennen typischerweise einzelne Worte und Zeichen voneinander. Die von Leerstellen bzw. Freiräumen getrennten Abschnitte werden vom linken Ende jeder Zeile aus verarbeitet und für die Erkennungsroutinen in einen Puffer übertragen.

Ein Zeichen-Erkennungsalgorithmus, Block **103**, verarbeitet die gepufferten Abschnitte und versucht, einzelne Zeichen zu erkennen. Wie genauer unter Bezugnahme auf den Zeichen-Erkennungsverarbeitungsabschnitt erläutert werden wird, werden unerkannte Zeilenabschnitte einer Anzahl von Verarbeitungsschritten unterworfen, um eine Zeichenerkennung zu erreichen. Ein "Delinierungs"-Verfahren lokalisiert und löscht Unterstreichungen und horizontale Striche für die Zeichenerkennung. Ein "Entschnörkelungs"-Prozeß scheidet Zeichen aus, welche durch gewundene Freiräume getrennt sind. Ein "Abschwächungs"-Prozeß löscht "Pfeffer und Salz"-Hintergrundstrukturen. Ein "Ausdünnungs"-Prozeß wirkt einer übermäßigen Komprimierung der Zeichen entgegen und macht die Zeichen erkennbar. Ein "Flick"-Prozeß bessert leicht unterbrochene Zeichenformen aus. Zeilenabschnitte, die auch danach noch unidentifiziert bleiben, werden in einem Zurückweisungs-Cachespeicher zur späteren Verarbeitung durch einen "Seitenanpaß"-Prozeß und einen Kontext-Analyseprozeß, Block **104**, gepuffert.

SEITEN-PARSING

Im folgenden wird auf **Fig.** 2(b) Bezug genommen, in der ein Ablaufdiagramm des Seiten-Parsingprozesses genauer dargestellt ist. Zweck des Seiten-Parsingprozesses ist die Annahme des Bit-mapped Seitenbildes als Eingabe und die Erzeugung einer geordneten Liste von Textblöcken als Ausgabe. Nach dem Abtasten einer Seite wird eine Bit-mapped Abbildung der Seite erzeugt. Aus dieser Bit-mapped Abbildung werden drei Felder erzeugt, Block **212**.

Bei dem bevorzugten Ausführungsbeispiel wird das Seitenbild mit einer Auflösung von 300 Punkten pro Zoll abgetastet, und der Parsing-Prozeß erzeugt seine Felder durch Analysieren jeder achten Abtastzeile des Bildes bzw. der Abbildung der Seite. Es wurde experimentell festgestellt, daß die Abtastung der Seite an jeweils der achten Abtastzeile (alle 8/300 eines Zolls bei dem beschriebenen Ausführungsbeispiel) eine ausreichend gute Auflösung zur Lokalisierung und Einordnung von besetzten Bereichen auf der Seite ergibt. Die Verwendung nur jeweils der achten Abtastzeile des Bildes reduziert die Prozeßzeit und die Speicheranforderungen beträchtlich. Es ist jedoch für den Fachmann klar, daß andere Abtastzyklen im Rahmen der Erfindung verwendet werden können.

Im folgenden wird auf **Fig.** 3(a) Bezug genommen, in der ein vergrößerter Bereich einer Seite **300** gezeigt ist. Der Bereich der Seite **300** stellt eine Zone einer Seite dar, die bei dem beschriebenen Ausführungsbeispiel in vierzig Zeilen abgetastet ist. Jedes der Quadrate, beispielsweise Quadrat **301**, stellt eine Zone von acht Abtastzeilen Höhe und acht Bits Breite dar. Der dargestellte Bereich der Seite **300** ist fünf Quadrate hoch und vier Quadrate breit.

In **Fig.** 3(b) ist ein Bit-mapped Bild desselben Bereichs einer Seite gezeigt. **Fig.** 3(a) und 3(b) sind charakteristisch für ein bei optischen Zeichen-Erkennungssystemen auftretendes Problem. Bei derartigen Systemen kann ein Buchstabe, beispielsweise der Buchstabe "O" **302** in **Fig.** 3(a) durch eine relativ grobe Annäherung des Buchstabens "O", z. B. durch

die Darstellung **312** in **Fig.** 3(b) [illegible]entiert werden. Darüberhinaus können Text und [illegible]hikelemente auf einer einzigen Seite vermischt sein. So ist beispielsweise das graphische Bild **303** mit dem Text in **Fig.** 3(a) vermischt. Die äquivalente Bit-mapped Zone ist bei **313** gezeigt. Es ist für den Fachmann klar, daß Bilder unter weiteren Problemen der Klarheit und der Schärfe als Folge der Transformation in eine digitale Bit-mapped Abbildung leiden können.

Ein durch den Seiten-Parsingprozeß des beschriebenen Ausführungsbeispiels erzeugtes erstes Feld ist ein horizontales Populationszählfeld. Jedes Element in diesem Feld enthält einen Zählwert der Zahl von Eins("1")-Bits aus vier aufeinanderfolgenden Bytes (32 Bits aus einer abgetasteten Scanzeile). Daher ist die durch jedes Element dieses Feldes dargestellte Zone 32 Bits breit und 8 Bits hoch. Bezugnehmend auf **Fig.** 3(b), haben die 32 Bits in der Abtastzeile **320** vierzehn 1-Bits. Daher wird der 8-Bit-hohe mal 32-Bit-breite Bereich in den Blöcken **331, 332, 333, 334** in dem horizontalen Populationszählfeld durch ein den Wert 14 enthaltendes Element dargestellt, dem Gesamtzählwert der 1-Bits in der Abtastzeile **320**. Die Abtastzeile **321** hat null 1-Bits, Zeile **322** hat neun 1-Bits, Zeile **323** hat fünf 1-Bits und Zeile **324** hat sechsundzwanzig 1-Bits, und jede dieser 8-Bit hohen mal 32-Bit breiten Zonen werden im Horizontalpopulationszählfeld mit entsprechenden Werten dargestellt.

Ein zweites Feld, das bei dem beschriebenen Ausführungsbeispiel der Erfindung verwendet wird, ist ein Vertikalpopulationszählfeld. Jedes Byte im Vertikalpopulationszählfeld enthält die Gesamtzahl von Eins ("1")-Bits in vier Bytes, wobei ein Byte aus jeder der vier aufeinanderfolgend abgetasteten Scan-Zeilen resultiert. Beispielsweise kann eine Eingabe im Vertikalpopulationszählfeld Byte **340, 341, 342** und **343** darstellen und einen Wert von 4 haben (Byte **340** hat zwei 1-Bits, Byte **341** hat null 1-Bits, Byte **342** hat zwei 1-Bits und Byte **343** hat null 1-Bits). Bei dem beschriebenen Ausführungsbeispiel enthält das Vertikalpopulationszählfeld eine Matrix, in der Zeilen der Matrix die Spalten des Bit-mapped Bildes und Matrixspalten Zeilen des Bit-mapped Bildes darstellen. Dies führt zu einer effizienten Verarbeitung bei der Implementierung des bevorzugten Ausführungsbeispiels.

Ein drittes Feld bzw. eine dritte Matrix bei dem Seiten-Parsingprozeß des beschriebenen Ausführungsbeispiels ist ein horizontales Phasenänderungsfeld. Jede Eingabe dieses Feldes stellt 32 Bits einer Abtastzeile dar, so daß das horizontale Phasenänderungsfeld die gleichen Abmessungen wie das horizontale Populationszählfeld hat. Jedes Feldelement enthält die Zählung der horizontalen Phasenänderungen (Übergänge zwischen Durchläufen von Einsen und Durchläufen von Nullen) in den 32 Bits. Die durch jedes Element des horizontalen Phasenänderungsfeldes dargestellte Zone ist 32 Bits breit und 8 Bits hoch. So haben beispielsweise die 32 Bits in Zeile **320** sieben Übergänge von Einsen zu Nullen oder von Nullen zu Einsen, die 32 Bits in Zeile **321** haben keine Übergänge, die 32 Bits in Zeile **322** haben acht Übergänge, die 32 Bits in Zeile **323** haben zwei Übergänge und die 32 Bits in Zeile **324** haben zwei Übergänge.

Auf der Basis der horizontalen Populationszähl- und horizontalen Phasenänderungsfelder wird ein Text-map-Feld erzeugt, Block **213**. Jedes Element des Text-map-Feldes stellt einen Bereich von 32 Bits Breite mal 8 Bits Höhe dar.

Im folgenden wird auf **Fig.** 4 Bezug genommen, in der ein Ablaufdiagramm zum Konstruieren des Text-map-Feldes dargestellt ist. Das Text-map-Feld wird durch einen Prozeß aufgebaut, der jede Spalte des Horizontalpopulationszählfeldes abwärts abtastet, um nach einem nicht-Null-Element zu suchen, Block **401**. Ein nicht-Null-Element im horizontalen Populationszählfeld zeigt das Vorhandensein von Text, Graphik oder Linien im entsprechenden Feld des Bit-mapped Bildes an. Nach Auffinden eines nicht-Null-Elements läuft der Prozeß in der Spalte des Horizontalpopulationszählfeldes abwärts und sucht nach einem Null-Element. Bei jedem verarbeiteten Element in einem Durchlauf von nicht-Null-Elementen werden die entsprechenden Horizontalphasenänderungszählungen summiert. Der Prozeß zählt auch die Gesamtanzahl von Elementen in einem Durchlauf von nicht-Null-Elementen, Block **402**.

Wenn die Zahl von Zeilen in einem Durchlauf größer oder gleich 2 und kleiner oder gleich 12 ist, Zweig **403**, und die Summe der Phasenänderungen größer oder gleich 8 und kleiner oder gleich 22 ist, Zweig **404**, hat die entsprechende Zone des Bit-mapped Bildes die Struktursignatur eines Textbereichs. Die Text-map-Feldelemente entsprechend jedem der Zeilenelemente im Durchlauf werden auf einen Code gesetzt, der angibt, daß eine Textzone vorliegt, Block **405**. Bei dem beschriebenen Ausführungsbeispiel werden diese Elemente auf einen Wert von TXTX gesetzt.

Wenn der Zeilenzählwert nicht größer als oder gleich 2 und kleiner als oder gleich 12 ist, Zweig **406**, wird eine Prüfung durchgeführt, um festzustellen, ob der Zeilenzählwert größer als oder gleich 24 ist, Zweig **407**. Wenn der Zeilenzählwert größer gleich 24 ist, enthält die entsprechende Zone des Bit-mapped Bildes eine Vertikallinie. Die Text-map-Feldelemente entsprechend den Horizontalpopulationszählelementen im Durchlauf werden auf einen Wert gesetzt, der eine Vertikallinie bezeichnet, Block **408**. Bei dem beschriebenen Ausführungsbeispiel werden diese Elemente auf einen Wert von TXVR gesetzt.

Wenn die Zeilenzählung kleiner als 2 oder zwischen 12 und 24 ist, Zweig **409**, ist dies ein Zeichen dafür, daß Graphik vorhanden ist. Die entsprechenden Text-map-Feldelemente werden auf einen Wert gesetzt, der das Vorhandensein von Graphik anzeigt, Block **410**. Bei dem beschriebenen Ausführungsbeispiel werden diese Elemente auf einen Wert von TXGR gesetzt.

Wenn die Zeilenzählung zwischen 2 und 12 liegt, der Phasenänderungszählwert jedoch entweder kleiner als 8 oder größer als 22 ist, Zweig **411**, wird ebenfalls das Vorhandensein von Graphik angezeigt. Die korrespondierenden Text-map-Feldelemente werden auf den Code eingestellt, der angibt, daß die entsprechenden Bits des Bit-mapped Seitenbildes Graphik enthalten, Block **412**.

Wenn das Ende des Horizontalpopulationszählfeldes noch nicht erreicht worden ist, Zweig **413**, wird die spaltenweise Verarbeitung fortgesetzt, wobei nach dem nächsten nicht-Null-Element gesucht wird, Block **401**. Anderenfalls wird der Prozeß beendet, Zweig **414**.

Es wurde experimentell festgestellt, daß der obige Prozeß ein ziemlich genaues Bild darüber ergibt, ob die Zonen der Bit-mapped Abbildung Text, Graphik oder vertikale Linien enthalten. Typischerweise treten die Zeichen in einem gewöhnlichen Text in einem weiten Höhenbereich auf, der normalerweise 2 bis 12 abgetastete Scannerzeilen einnimmt. Daher prüft der oben beschriebene Prozeß Durchlaufzeilenzählwerte von nicht-Null-Elementen im Bereich von 2 bis 12 Zeilen.

Es wurde außerdem experimentell festgestellt, daß die Gesamtanzahl von Phasenänderungen in einem vorgegebenen Durchlauf von nicht-Null-Elementen über einen Zeichengrößenbereich von angenähert 4 bis 24 Punkten im wesentli-

chen konstant bleibt, obwohl Zeichen mit größeren Höhen weniger Phasenänderungen pro Abtastung als Zeichen geringerer Höhe haben. Daher ist bei dem beschriebenen Ausführungsbeispiel ein Phasenänderungsgesamtzählwert bei einem Durchlauf von nicht-Null-Elementen im horizontalen Populationszählfeld zwischen 8 und 22 ein Indikator für gedruckten Text.

Der Seiten-Parsing-Prozeß macht dann einen anderen Durchlauf und versucht, einen relativ großen Text zu lokalisieren, Block **214**. Der Vorgang zum Lokalisieren eines großen Textes läuft im wesentlichen ebenso wie der zuvor beschriebene Vorgang des Aufbaus des Text-map-Feldes ab, ausgenommen, daß der Vorgang zum Lokalisieren eines großen Textes jede vierte Scanzeile und jedes vierte Element in dem Phasenänderungsfeld prüft. Daher sucht die Routine für großen Text in dem Bit-mapped Bild mit 1/4 der Auflösung des Prozesses zum Lokalisieren eines Textes normaler Größe und identifiziert Text bis zu einer vierfachen Größe des Normaltextes. Bei dem beschriebenen Ausführungsbeispiel ist der größte, mit dieser Routine auffindbare Text 48 Abtastzeilen hoch. Bei 300 Abtastzeilen pro Zoll entspricht dies einem Text einer Höhenabmessung von 3,25 cm oder etwa 92 Punkten. Text-Map-Feldzellen entsprechend den Zonen in der Bit-mapped Seitenabbildung, in denen ein großer Text mit dem zuvor beschriebenen Prozeß gefunden wurde, werden auf einen Wert eingestellt, der einen Großtextinhalt kennzeichnet. Bei dem beschriebenen Ausführungsbeispiel ist dieser Wert TXTH.

In dem Seiten-Parsing Prozeß wird versucht, Blöcke mit nur-Text von ganz weißen Blöcken, Graphik enthaltenden Blöcken oder Blöcken mit vertikalen Linien zu trennen. Als Teil dieses Prozesses wird bei dem beschriebenen Ausführungsbeispiel in einem Seiten-Parsing Prozeß versucht, den ungefähren Schrägfehler bzw. Versatz (skew) der Seite zu bestimmen und das Text-Map-Feld sowie das Vertikalpopulationszählfeld auf einen derartigen Schrägfehler einzustellen, Block **215**.

Im folgenden wird auf **Fig. 5** Bezug genommen, in der ein Ablaufdiagramm des bei dem beschriebenen Beispiel benutzten Verfahrens zur Entzerrung einer Seite dargestellt ist. Zunächst werden Gesamtschrägfehler- und Abtastzählungsvariable auf Null initialisiert, Block **501**.

Im Text-Map-Feld wird danach jede Spalte abwärts abgetastet, um nach zusammenhängenden Textzellen zu suchen, Block **502**. Wenn zusammenhängende Textzellen lokalisiert sind, wird eine erste Variable, die benutzt wird, wenn Zellen links der derzeitigen Textzelle (LEFT) geprüft werden, auf –1 gesetzt. Eine zweite Variable, die benutzt wird, wenn Zellen rechts der derzeitigen Textzelle (RIGHT) geprüft werden, wird auf 1 gesetzt, Block **503**.

Für jede Textzelle, die der ersten Textzelle im Durchlauf folgt, werden Zellen links und rechts der Textzelle im Text-Map-Feld geprüft. Wenn die Zelle links leer ist (d. h. die Zelle ist nicht als Text-, Graphik- oder Vertikallinien enthaltende Zelle angegeben, wobei die o. g. TXTX, TXVR, TXTH oder TXGR-Codes verwendet werden), so wird die Gesamtschrägfehler-Variable um den Istwert von LEFT inkrementiert, und die Abtastzählvariable wird um 1 inkrementiert. Ist die Zelle besetzt (enthält sie TXTX, TXVR, TXTH oder TXGR), so wird LEFT auf 1 gesetzt und die Gesamtschrägfehler-Variable und die Abtastzählwert-Variable werden nicht modifiziert, Block **504**. Wenn die Zelle im Text-Map-Feld rechts von der derzeitigen Zelle leer ist, so wird die Gesamtschrägfehler- bzw. Gesamtverzerrungsvariable um den Wert von RIGHT inkrementiert, und die Abtastzählwert-Variable wird um 1 inkrementiert. Wenn die Zelle besetzt ist, wird RIGHT of –1 gesetzt, und die Gesamtschrägfehler-Variable und Abtastzählwert-Variable werden nicht modifiziert, Block **505**.

Wenn mehrere Zellen beim Durchlauf der Textzellen vorhanden sind, werden die Blöcke **504** und **505** für diese weiteren Zellen wiederholt, Zweig **506**. Anderenfalls, wenn mehrere Zellen im Text-Map-Feld sind, Zweig **507**, werden sie in einem anderen Durchlauf der Textzellen abgetastet, Zweig **508**. Nach dem Prüfen der gesamten Textabbildung (Text-Map), Zweig **509**, wird ein angenäherter Schrägfehler für die Seite berechnet, und das Text-Map-Feld und das vertikale Populationszählfeld werden verschoben, um die berechnete Verzerrung bzw. den berechneten Schrägfehler zu kompensieren, Block **510**. Der angenäherte Schrägfehler wird dadurch berechnet, daß man die Gesamtschrägfehler-Variable durch die Abtastzählung teilt. Dieser Vorgang führt zu einer Annäherung des Schrägfehlers für die Seite als Anzahl der Zeilen pro 4096 Spalten. Die Text-Map- und Vertikalpopulationszähl-Felder werden danach ausgerichtet.

Bei dem oben beschriebenen Verfahren der Schrägfehlerjustierung wird unterstellt, daß der Text auf einer Seite generell in horizontalen Zeilen auf der Seite angeordnet ist. Das Text-Map-Feld, das erfindungsgemäß gewonnen wird, hat eine gute Vertikalauflösung (bis 1/38 eines Zolls). Wenn eine Seite einen Schrägfehler hat, so treten Leerzellen an Stellen auf, die eigentlich Textzellen einnehmen müßten. Das oben beschriebene Verfahren macht diese Annahmen nutzbar, um den angenäherten Schrägfehler zu bestimmen.

Nach der Einstellung einer angenäherten Schräglage werden horizontale und vertikale Verläufe weißer Stellen im Text lokalisiert, Block **216**. Zweck der Lokalisierung dieser Leerstellenverläufe ist die Trennung von Textblöcken. Bei dem beschriebenen Ausführungsbeispiel wird das entzerrte Text-map-Feld nach horizontalen Leerstellenverläufen bzw. horizontalen weißen Zwischenräumen geprüft (Leerstellen oder weiße Räume können als Zellen im Text-map-Feld definiert werden, die nicht mit TXTX, TXVR, TXTH oder TXGR codiert sind). Das erfindungsgemäße Verfahren betrachtet eine Zone als horizontalen Weg, wenn ein weißer oder freier Raum existiert, der wenigstens 8 Pixel breit und 192 Pixel in Horizontalrichtung der Seite lang ist. In ähnlicher Weise werden vertikale Leerraumverläufe oder Wege unter Verwendung des Text-map-Feldes lokalisiert. Bei dem bevorzugten Ausführungsbeispiel der Erfindung wird eine Zone als vertikaler Weg angesehen, wenn er einen Weißraum- oder Leerraumbereich von wenigstens 16 Pixel Breite und 192 Pixeln Länge in der Vertikalrichtung der Seite hat. Die oben genannten Längen- und Breitenabmessungen für horizontale und vertikale Wege wurden auf experimentellem Wege von der Anmelderin bestimmt, um horizontale und vertikale Wege durch gedruckten Text zu lokalisieren.

Im folgenden wird auf **Fig. 6** Bezug genommen, in der ein Ablaufdiagramm eines Verfahrens zur Lokalisierung von Wegen durch den Text dargestellt ist. Als erster Schritt bei der Lokalisierung von Wegen wird ein Verfahren verwendet, durch das obere, untere, linke und rechte Ränder der Seite effektiv ignoriert werden, Block **601**.

Bei dem bevorzugten Ausführungsbeispiel der Erfindung erfolgt eine Maskierung der Ränder durch Erzeugung einer Version des Vertikalpopulationszählfeldes, bei der jede Zelle des Vertikalpouplationszählfeldes eine 32 × 32 Pixel-Quadratzone des ursprünglichen Bit-mapped Bildes darstellt. Effektiv stellt jede Zelle in dieser Version des Vertikalpopula-

tionszählfeldes vier aufeinander[redacted]de Zellen in einer Spalte des ursprünglichen [redacted]kalpopulationszählfeldes dar. Die komprimierte Version des Vertikalpopulationszählfeldes wird sodann "verwischt".

Diese Technik wird bei der Erfindung benutzt, um ein Bild in einer Anzahl von Richtungen zu verschieben und eine logische ODER-Operation an dem ursprünglichen Bit-mapped Bild und dem verschobenen Bit-mapped Bild durchzuführen. Im Ergebnis expandiert diese Methode Text oder Graphik enthaltende Bereiche, während die dazwischenliegenden Räume und Grenzbereiche enger gemacht werden. Die restlichen weißen Randbereiche werden für das Weg-Auffindungsverfahren im Vertikalpopulationszählfeld als nicht-verfügbar markiert. Diese Methode wird unter Bezugnahme auf die Beschreibung der Zeichenerkennung weiter unten genauer beschrieben.

Danach wird ein Feld, genannt Weg-map-Feld, zur Speicherung der horizontalen und vertikalen Weginformationen erzeugt, Block **602**. Das Weg-map-Feld hat dieselbe Auflösung wie das Text-map-Feld. Jede Spalte stellt 32 Bits der aufgenommenen Scanlinie dar und jede Zeile stellt eine Abtastung bei jeder achten Scanzeile des Bildes dar. In dem Weg-map-Feld wird die Existenz eines horizontalen Weges durch Setzen der Bits in einem Eintrag im Feld angezeigt. Das Vorhandensein eines vertikalen Weges wird durch Setzen eines anderen Bits im Eintrag gekennzeichnet.

Bei dem beschriebenen Ausführungsbeispiel wird als erster Schritt bei dem Aufbau des Weg-map-Feldes für ein Setzen der am weitesten links und rechts gelegenen Spalten und der obersten und untersten Zeilen des Weg-map-Feldes gesorgt, um das Vorhandensein von Wegen anzuzeigen. Dieser Schritt stellt sicher, daß ein Weg, der in den Rand einer Seite mündet, einen Block erzeugt. Bei vollständiger Besetzung umrandet das Weg-map-Feld sowohl Text- als auch Graphikblöcke mit horizontalen und vertikalen Wegen (Freiräumen).

Nach der Erzeugung der Weg-map und der Initialisierung der am weitesten links und rechts gelegenen Spalten und obersten und untersten Zeilen der Weg-map werden horizontale Wege erzeugt, Block **603**. Jede Zeile des Text-map-Feldes wird nach aufeinanderfolgenden leeren Einträgen (d. h. Einträgen, die nicht auf TXTX, TXVR, TXTH oder TXGR gesetzt sind) abgetastet. Wenn eine Folge eine Länge von wenigstens einer vorgegebenen Anzahl von Bytes hat, werden die horizontalen Wegbits in den entsprechenden Einträgen im Weg-map-Feld gesetzt. Bei dem beschriebenen Ausführungsbeispiel ist die vorgegebene Anzahl von Bytes 6. Jeder horizontale Weg, der unter Verwendung dieser Methode gefunden wird, wird sowohl um eine Spalte nach links von dem horizontalen Weg als auch um eine Spalte nach rechts von dem horizontalen Weg ausgedehnt. Dies stellt sicher, daß die horizontalen Wege an den Blockrändern in vertikale Wege selbst dann übergehen, wenn sie aus Graphiken oder Kopfzeilen innerhalb von 32 Pixeln links oder rechts vom Rand des Blocks bestehen.

Danach wird das Text-map-Feld abgetastet, um Vertikallinien aufzufinden (d. h. Einträge, welche auf TXVR gesetzt sind), Block **604**. Die entsprechenden Eingaben im Weg-map-Feld haben ein gesetztes Bit, das anzeigt, daß es eine vertikale Linie an dieser Stelle in der Abbildung gibt. Dieses Bit wird bei einer späteren Verarbeitung geprüft, bei der relativ enge Wege aus dem Weg-map-Feld eliminiert werden.

Als nächstes werden vertikale Wege in dem Weg-map-Feld erzeugt, Block **605**. Jede Zeile des Vertikalpopulationszählfeldes wird nach aufeinanderfolgenden leeren Einträgen (entries) abgetastet. Wenn eine Folge wenigstens eine vorgegebene Anzahl von Einträgen lang ist, wird sie als möglicher Vertikalweg angesehen. Bei dem beschriebenen Ausführungsbeispiel ist die vorgegebene Anzahl gleich 6. Wenn ein Ende eines Vertikalweges nicht mit einem Horizontalweg zusammenfällt, wird der Vertikalweg abgebrochen, bis er an beiden Enden mit einem Horizontalweg zusammenfällt. Vertikalwege erstrecken sich stets von einer Schnittstelle mit einem Horizontalweg zu einer Schnittstelle mit einem anderen Horizontalweg.

Die Weg-map wird danach geprüft, und alle Vertikalwege, die nur einen Eintrag breit sind, werden entfernt, sofern der Eintrag nicht angibt, daß der Vertikalweg wegen einer entsprechenden Vertikallinie im Bild bzw. in der Abbildung gesetzt wurde. Die Weg-map wird dann wiederum abgetastet, und Abschnitte der horizontalen Wege werden gelöscht, soweit sie nicht in einem Vertikalweg beginnen und enden, Block **606**.

Eine Block-Lokalisierungsroutine, Block **217**, benutzt zuvor durch den Seiten-Parsing-Prozeß gebildete Datenstrukturen, beispielsweise das Text-map-Feld, und bildet zwei neue Datenstrukturen: ein Block-map-Feld und eine Blockliste.

Im folgenden wird auf **Fig. 7** Bezug genommen. Das Block-map-Feld **701** enthält ein Feld mit den gleichen Abmessungen, wie das Text-map-Feld. Jede 1-Byte-Zelle im Block-map-Feld, beispielsweise die Zelle **702**, enthält eine Blockzahl für diese Zelle. Zellen, die noch nicht als Teil eines Blockes bezeichnet worden sind, werden bei dem beschriebenen Ausführungsbeispiel mit einer 0 gekennzeichnet. Dabei kann es ein Maximum von 255 Blöcken pro Seitenabbildung geben. Die Blockzahl, beispielsweise die Blockzahl 1 an der Zelle **702**, ist ein Hinweis in eine Blockliste **703**. Jeder Eintrag in der Blockliste enthält Informationen über jeden Block, z. B. Koordinaten des Blocks, Spalte **705**, Zellenzählinformationen, Spalte **706**, und Abstand zu anderen Blöcken, Spalte **707**. Die Koordinateninformationen **705** enthalten Informationen über die obersten, untersten, am weitesten links und am weitesten rechts gelegenen Pixel. Die Zellenzählinformation **706** enthält Informationen über die Anzahl von Textzellen, die Anzahl großer Textzellen und die Anzahl graphischer Zellen innerhalb der Grenzen des Blocks.

Gemäß **Fig. 8**, auf die jetzt Bezug genommen wird, enthält der Vorgang des Lokalisierens von Blöcken, d. h. der Vorgang **217** aus **Fig. 2**, die Schritte einer Block-Auffindungsroutine, wobei zunächst die Spalten der Weg-map abgetastet werden, Block **801**. Die Block-Auffindungsroutine tastet jede Zelle der Weg-map nach Zellen ab, welche weder von horizontalen noch von vertikalen Wegen durchquert werden. Nach dem Auffinden einer Zelle, die weder von horizontalen noch von vertikalen Wegen durchquert wird, prüft die Block-Auffindungsroutine die entsprechende Zelle im Block-map-Feld. Wenn die entsprechende Zelle im Block-map-Feld unbesetzt ist, d. h. der derzeitige Wert eine 0 ist, so ruft die Block-Auffindunsroutine eine Block-Zerlegungsroutine auf, Block **802**. Die Block-Zerlegungsroutine prüft das Weg-map-Feld der untersuchten Zelle zur Ermittlung vertikaler Wege auf den linken und rechten Seiten dieser Zelle. Die Block-Zerlegungsroutine tastet danach Zeilen des Weg-map-Feldes ab. Für jede Zeile lokalisiert die Block-Zerlegungsroutine vertikale Wege auf der linken und rechten Seite jeder Zelle über der laufenden Zelle. Wenn die Block-Zerlegungsroutine eine Zelle lokalisiert, deren linker oder rechter Rand entsprechend dem Verlauf der Vertikalwege von dem linken oder rechten Rand der laufenden Zelle um mehr als eine Zelle differiert, so erzeugt die Block-Zerlegungsroutine einen horizontalen Weg an dieser Zelle. Die Block-Zerlegungsroutine arbeitet sich in ähnlicher Weise in der Spalte der

laufenden Zelle abwärts, u[illegible]e unterste Zeile des derzeitigen Blocks zu bestim[illegible]. Dieser Vorgang erzeugt einen etwa rechteckigen Block, der mit der weiter unten beschriebenen erfindungsgemäßen Methode weiter verarbeitet wird.

Nachdem die Block-Zerlegungsroutine die linken, rechten, oberen und unteren Ränder eines etwa rechteckigen Zellenblocks bestimmt hat, schließt sich eine Block-Statistikroutine an, Block **803**, die zum Markieren anderer als zu dem gleichen Block gehöriger Zellen verwendet wird. Die Block-Statistikroutine markiert jede Zelle im Block-map-Feld, die von den linken, rechten, obersten und untersten Wegen des aktuellen Blocks als zum aktuellen Block gehörig begrenzt sind. Außerdem zählt die Block-Statistikroutine die Anzahl von Text- und Graphikzellen im aktuellen Block durch Prüfung des Text-map-Feldes. Die relative Anzahl von Textzellen gegenüber Graphikzellen dient zur Feststellung, ob der Block als Text- oder Graphikblock für die Weiterverarbeitung zu klassifizieren ist. Die Block-Statistikroutine sammelt außerdem Informationen über die mittlere Länge von Folgen belegter Zellen im Text-map-Feld für jede Spalte im Block. Diese Information dient zur Feststellung der angenäherten Höhe der Zeichen in den Textblöcken. Die Block-Statistikroutine bildet außerdem ein Histogramm der Anzahl der Folgen von belegten Zellen in den Spalten des Text-map-Feldes. Der Medianwert in diesem Histogramm ist eine Näherung für die Anzahl von Textzeilen in dem Block. Die Block-Statistikroutine berechnet auch die extrem linken, rechten, obersten und untersten Koordinaten des Blocks. Wie oben gesagt, werden die oben angegebenen Koordinaten-, Zählwert- und Statistikinformationen in der Blockliste gespeichert, Block **703**.

Wie oben gesagt, läßt die hier verwendete Blockliste nur 255 Einträge zu. Die Anzahl von Einträgen ist bei dem beschriebenen Ausführungsbeispiel beschränkt, um eine gewisse Verarbeitungseffizienz zu ermöglichen. Es ist jedoch für den Fachmann klar, daß eine andere Anzahl von Einträgen in der Blockliste benutzt werden kann, ohne vom Erfindungsgedanken abzuweichen. Bei dem beschriebenen Ausführungsbeispiel wird der Prozeß mit einer größeren Vertikalwegbreite zur Bestimmung der Blockgrenzen wiederholt, wenn mehr als 255 Blöcke unter Verwendung des zuvor beschriebenen Prozesses ermittelt werden. Es ist außerdem klar, daß die Erfindung eine andere Breite für die horizontalen und vertikalen Wege vorsehen könnte.

Nach Beendigung der Verarbeitung eines ersten Blocks wird die Block-Auffindungsroutine, Block **801**, fortgesetzt, wenn weitere Zellen zur Verarbeitung übrig bleiben, Zweig **804**. Nach der Beendigung der Verarbeitung aller Zellen im Block-map-Feld, Zweig **806**, ist der Vorgang der Blocklokalisierung beendet.

Danach werden die Blöcke gruppiert, Block **218**. Die Blöcke werden entsprechend ihrer Relativanordnung, ihrer Inhalte (Graphik oder Text) und ihrer Textur bzw. Beschaffenheit (Schriftgröße und Zeilenabstand) geordnet. Die Blockgruppierungsinformation wird in einer Gruppenliste aufgezeichnet. Jedem Block in der Blockliste wird eine Gruppennummer zugeordnet. Die Gruppennummer dient als Index für die Gruppenliste.

Für jeden Block in der Blockliste wird das Block-map-Feld über, unter, links und rechts von dem Block abgetastet. Blöcke, die von unterschiedlicher Art oder Beschaffenheit bzw. Struktur sind, werden nicht zusammengruppiert. Außerdem werden solche Blöcke nicht zusammengefaßt, die mehr als einen vorgegebenen Vertikalabstand oder mehr als einen vorgegebenen Horizontalabstand entfernt voneinander angeordnet sind.

Im folgenden wird auf **Fig. 9** Bezug genommen, in der ein Beispiel für die Blockgruppierung gezeigt ist. Beispielsweise sind die Blöcke 1, 6 und 11 als Gruppe 1, **901**, zusammengefaßt. In dem speziellen Beispiel können diese Blöcke beispielsweise eine Überschrift für die Seite enthalten. Eine Überschrift ist häufig durch höhere Schrifttypen als der Rest des Textes auf der Seite unterschieden. Wegen der unterschiedlichen Schriftgröße werden diese Blöcke zusammengruppiert und nicht den restlichen Blöcken auf der Seite zugeschlagen.

Blöcke 2, 3, 4 und 5 sind als Gruppe 2, **902**, zusammengruppiert. Das erfindungsgemäße Verfahren prüft Blöcke, die einem aktuellen Block benachbart sind, um festzustellen, ob der Horizontalabstand H **903** größer als sein vorgegebener Wert ist. Bei dem beschriebenen Beispiel ist dieser vorgegebene Wert 6 Zellenspalten. Da der Horizontalabstand H **903** zwischen Block 2 und Block 7 größer als der vorgegebene Horizontalabstandsgrenzwert ist, sind Blöcke 7 und 8 nicht mit Gruppe 1 zusammengruppiert.

Blöcke 7 und 8 sind als Gruppe 3 zusammengefaßt, **904**. Blöcke 9 und 10 sind separat von den Blöcken 7 und 8 gruppiert und als Gruppe 4, **906**, bezeichnet, da der Vertikalabstand **905** zwischen Block 8 und Block 9 einen vorgegebenen Grenzwert übersteigt. Bei dem beschriebenen Beispiel ist die vorgegebene Grenze für den Vertikalabstand zwischen Blöcken 12 Zellenzeilen. Blöcke 12, 13, 14 und 15 sind als Gruppe 5, **907**, zusammengruppiert. Block 16 ist separat als Gruppe 6, **909** gruppiert. Block 16 ist nicht mit Blöcken 12, 13, 14 und 15 gruppiert, da es einen Graphikblock **908** gibt.

Die Erfindung ermöglicht außerdem die Feststellung von Rändern von Spalten auf einer Seite durch Prüfen aufeinanderfolgender Blöcke in Seiten-Abwärtsrichtung, wodurch festgestellt wird, ob der linke Rand jedes Blocks angenähert mit dem darunterliegenden Block ausgerichtet ist und ob der Block vertikal innerhalb eines vorgegebenen Abstands von dem Nachbarblock liegt. Wenn der Block nicht angenähert ausgerichtet mit dem darunterliegenden Block ist, oder der Block nicht den vorgegebenen Abstand von seinem Nachbarn hat, wird angenommen, daß die Blöcke nicht in einer Spalte liegen.

Nachdem die Blöcke gruppiert worden sind, wird das Block-map-Feld unter Verwendung der Gruppennummern anstatt der Blocknummern in jedem Element des Block-map-Feldes wieder aufgebaut. Dadurch reduzieren sich die nachfolgenden Verarbeitungserfordernisse.

Nach der Beendigung der Blockgruppierung werden die Blöcke für die Ausgabe an die Zeilen-Parsingroutinen, Block **219**, geordnet. Der Zweck der Anordnung der Gruppen für die Ausgabe besteht darin, daß Gruppen an die Zeilen-Parsingroutinen in der logischen Ordnung ausgegeben werden, in der sie auch gelesen würden. Im folgenden wird auf **Fig. 10(a)** Bezug genommen, in der ein Blockdiagramm mit dem Seitenbild bzw. der Seitenabbildung, bestehend aus 7 Textblöcken, gezeigt ist. Die Seitenabbildung enthält eine Kopf- oder Überschriftzone **1001** und 3 logische Textspalten.

Als erster Schritt bei der Anordnung von Gruppen für die Ausgabe werden die vertikal benachbarten Blockgruppen lokalisiert. Unter Verwendung der die Lokalisierung von vertikal benachbarten Blöcken betreffenden Informationen wird ein Baum konstruiert, der die Blöcke verknüpft. Jeder Verzweigungspunkt im Baum stellt einen Text- oder Graphikblock dar und enthält Hinweise auf bis zu 8 Blöcke über ihm. Wenn mehr als ein Block über dem aktuellen Block ist, sind die Hinweise so angeordnet, daß Blöcke von links nach rechts geordnet werden. Die Wurzel des Baums ist am Seitenende.

Jedem Block ist ein Verzweigungspunkt auf der Basis einer links-nach-rechts-Baumdurchquerungsreihenfolge zugeordnet. Verzweigungspunkt 0 ist die Wurzel.

Wie in **Fig.** 10(a) dargestellt ist, bildet der Knoten 1 1011 eine erste Vertikalgruppe **1002**. Knoten 2 1001, Knoten 3 1013, Knoten 4 1014 und Knoten 5 1015 können eine zweite Vertikalgruppe **1003** bilden, die im wesentlichen der ersten Vertikalgruppe benachbart ist. Knoten 6 1016 und Knoten 0 1010 können eine dritte Vertikalgruppe **1004** bilden, die der zweiten Vertikalgruppe im wesentlichen benachbart ist.

Um die Ausgabereihenfolge der Blöcke zu bestimmen, wird der Baum von links nach rechts durchlaufen, und jedem Zweig des Baums wird bis zu dessen Ende gefolgt, wobei jeder Nebenast von links nach rechts verfolgt wird. Generell wird ein Knoten am Ende eines Zweiges zuerst ausgegeben, und Knoten von jedem Zweig eines Unterbaums werden ausgegeben, bevor der Wurzelknoten für diesen Unterbaum ausgegeben wird.

So wird unter Bezugnahme auf **Fig.** 10(a) die normale Ausgabefolge für die Blöcke 0 bis 6 dadurch bestimmt, daß der Baum vom Block 0 (der Wurzel), 1010, aus zum ersten Knoten auf der linken Seite, Block 5, 1015, durchschritten wird. Vom Block 5 1015 abzweigende Knoten werden danach von links nach rechts durchschritten. Daher ist Block 1 1011 der nächstgeprüfte Knoten. Da keine Blöcke vom Block 1 1011 abzweigen, ist dieser Block als der erste Block bezeichnet, der vom Baum abgetrennt und zu den Zeilen-Parsingroutinen geschickt werden soll. Der nächste vom Block 5 1015 abzweigende Knoten ist Block 4 1014. Daher wird Block 4 1014 als nächster verarbeitet. Block 4 1014 hat Zweige. Daher wird er durchschritten und als nächstes wird Block 2 1001 geprüft, da er vom Block 3 1013 abzweigt. Wenn es keine anderen Blöcke gibt, die vom Block 2 1001 abzweigen, ist Block 2 1001 die nächste Block-Ausgabe an die Zeilen-Parsingroutinen. Block 3 1013, der keine Zweige hat, ist der nächste, an die Zeilen-Parsingroutinen auszugebende Block, gefolgt vom Block 4, 1014. Da es keine weiteren Blöcke gibt, die vom Block 5 1015 abzweigen, ist dieser Block die nächste Blockausgabe an die Zeilen-Parsingroutinen. Der Wurzelknoten 1010 wird weiterhin von links nach rechts durchlaufen, und Block 6 1016 wird verarbeitet. Da auch hier keine Blöcke vom Block 6 1016 abzweigen, ist letzterer der nächste Block, der zu den Zeilen-Parsingroutinen geschickt wird. Schließlich gibt es keine weiteren von dem Wurzelknoten 1010 abzweigenden Blöcke, und der Wurzelknoten wird zu den Zeilen-Parsingroutinen übertragen.

Bei der Blockverarbeitung werden die als Graphikblöcke bezeichneten Blöcke zwar in den oben beschriebenen Durchlauf- und Sortierprozeß eingebunden, jedoch nicht zu den Zeilen-Parsingroutinen übertragen.

Bei Verwendung des oben beschriebenen Verfahrens zur Blockausgabe an den Zeilen-Parser sind gewisse Seitenlayouts besonders fehleranfällig. So können beispielsweise in dem Seitenlayout gemäß **Fig.** 10(a) die Blöcke 2 und 3 Kopfetiketten sein. In einem solchen Fall würde die logische Lesefolge der Seite von den aus dem oben beschriebenen Prozeß gewonnenen Resultaten abweichen. Daher wird die Ausgabereihenfolge von Blöcken bei dem beschriebenen Ausführungsbeispiel der Erfindung durch eine rekursiv aufrufende Routine umgeordnet. Die Routine wird für jeden Knoten mit mehr als einem Aufwärtszweig aufgerufen. Bei dem Beispiel gemäß **Fig.** 10(a) würde beispielsweise die Routine aufgerufen, wenn Block 5 1015 und der Wurzelblock 1010 verarbeitet werden.

Die Rekursivroutine findet den oberen Block jedes Zweiges. Beginnend mit dem am weitesten links gelegenen Zweig prüft die Routine die Knoten an dem nächsten Zweig rechts. Wenn das obere Ende eines Knotens des rechts gelegenen Zweiges auf der Seitenabbildung höher als die obere Zeile eines Knotens des linken Zweiges ist (entweder überlappt der rechte Knoten den linken Knoten, oder das untere Ende des rechten Knotens liegt oberhalb des oberen Endes des linken Knotens), so wird der Unterbaum in dem rechten Zweig auf den linken Zweig verpflanzt. Dieser Vorgang wird für jeden Knoten wiederholt, der die o. g. Kriterien erfüllt.

So liegen beispielsweise sowohl Knoten 2 1001 als auch Knoten 3 1013 über dem Knoten 1 1011, so daß Knoten 3 1013 für die Zwecke der Ausgabe an die Block-Parsingroutinen auf Knoten 1 1011 umgepflanzt wird. Der nach der Bearbeitung durch diese Rekursivroutine gewonnene Baum ist in **Fig.** 10(b) gezeigt. In **Fig.** 10(b) wurden die Blocknummern neu zugeordnet, und zwar in einer links nach rechts-Baum-Durchlaufreihenfolge. Die neuen Blocknummern zeigen die Reihenfolge der Ausgabe an die Seiten-Parsingroutinen. Die Blöcke werden an die Seiten-Parsingroutinen ausgegeben, beginnend mit Block 1 1001, danach Block 2 1013, Block 3 1011, Block 4 1014, Block 5 1015, Block 6 1016 und schließlich der Wurzelknoten, Block 0 2020. Die logische Verknüpfung zwischen den Blöcken wurde am Zweig **1020** geändert.

Eine Anzahl abschließender Einstellungen werden als Teil des Seiten-Parsing-Prozesses durchgeführt. Diese Einstellungen umfassen Prozesse zum weiteren Verschmelzen benachbarter Blöcke, nachdem die Block-Ausgabereihenfolge bekannt ist, Vergrößern von Blöcken bis hinein in den den Block umgebendenen weißen Raum, Konstruieren eines Modells der Seitenabbildung, das die Lage und Reihenfolge der Textblöcke zeigt, neues Schrägstellen der Block-map- und der Wege-map-Felder zur Umkehr der Verzerrungskorrektur des früheren Prozesses und den Aufbau eines Blockdeskriptorfeldes mit beschreibenden Informationen über jeden Block.

Wesentlich ist, daß das Modell der Seitenabbildung mit der Platzierung und Reihenfolge der Textblöcke auf einem Gerät, beispielsweise an einem Graphikterminal von dem Benutzer sichtbar gemacht werden kann. Der Benutzer kann dann die Ausgabereihenfolge der Blöcke wahlweise ändern. Dieses Verfahren ermöglicht dem Benutzer die Korrektur der Ausgabereihenfolge der Blöcke, soweit die Seiten-Parsingroutinen unrichtige Annahmen bezüglichen der logischen Lesefolge der Seite gemacht haben.

BLOCK-PARSING

Jeder von der Seiten-Parsingroutine erzeugte Block wird an die Block-Parsingroutine in der gewünschten Reihenfolge weitergeleitet. Die Block-Parsingroutine versucht, jeden Block in einzelne Textzeilen zu zerlegen. Die Block-Parsingroutinen verwenden von den Seiten-Parsingroutinen und dem Bit-mapped Bild des Eingangsblocks aufgebaute Datenstrukturen zum Isolieren einzelner Zeilen und zum Hinzufügen von Daten zu der Block-Beschreibungsinformation in der Blockliste. Die zur Blockliste hinzugefügten Daten enthalten Informationen, welche die äußerst linke Spalte im Block, die Breite des Blocks, die Höhe des Blocks, die Anzahl von Zeilen im Block und die Startzeilennummer identifizieren.

Nach Empfang eines Eingangsblocks berechnet die Block-Parsingroutine den Schrägfehler des Blocks, Block **221** der

Fig. 2(c). Der Schrägfehl[illegible]s Blocks wird auf der Basis einer detaillierten A[illegible]se der Phasenänderungszählungen in dem Horizontalphasenänderungsfeld berechnet. Als nächstes werden einzelne Zeilen isoliert, Block **222**, und zwar durch Prüfen des Bit-mapped-Bildes für den Block in Verbindung mit der Phasenänderungszählungsanalyse, um die Lage des möglicherweise schräggestellten horizontalen weißen Raums zu bestimmen, der die Zeilen voneinander trennt.

Der Block-Parsing-Prozeß trennt und zerschneidet die Zeilen, Block **223**, durch Lokalisierung von Horizontalwegen des geringsten Widerstandes, die der berechneten Schräglage am nächsten kommen. Der im wesentlichen horizontale Wege, der Textzeilen trennt, kann von Buchstaben unterbrochen sein, die Unter- oder Oberlängen haben. Beispielsweise haben die Kleinbuchstaben "g", "j", "p", "q" und "y" alle Unterlängen. Die Block-Parsingroutine schneidet um solche Buchstaben herum, um sicherzustellen, daß die Unterlängen derartiger Buchstaben in der richtigen Zeile zurückbleiben, wenn die Zeile an die Zeilen-Parsingroutinen weitergegeben wird.

Immer wenn die Block-Parsingroutine nicht in der Lage ist, ein Hindernis innerhalb gegebener Toleranzen zu vermeiden oder zu umfahren, mißt die Block-Parsingroutine die Abmessungen des Hindernisses, um festzustellen, ob das Hindernis eine Vertikallinie ist. Wenn das Hindernis eine Vertikallinie ist, wird es gelöscht. Wenn das Hindernis keine Vertikallinie ist, durchschneiden die Block-Parsingroutinen das Hindernis. Einzelzeilen werden getrennt und für die Weiterverarbeitung durch die Zeilen-Parsingroutinen gepuffert, Block **224**.

ZEILEN-PARSING

Jede Zeilenausgabe aus den Block-Parsingroutinen dient als Eingabe für die Zeilen-Parsingroutinen. Die Zeilen-Parsingroutinen suchen eine Textzeile in Einzelbuchstaben zu zerlegen. Bezugnehmend auf **Fig. 2(d)** finden die Zeilen-Parsingroutinen bei dem beschriebenen Ausführungsbeispiel zunächst alle Spalten, die einen weißen Raum von der Oberseite der Zeile zur Unterseite der Zeile haben, Block **231**.

Die Spalten oder Segmente, die einen weißen Raum von der Zeilenoberseite zur Zeilenunterseite haben, werden danach getrennt und gerahmt, Block **232**. Um Segmente zu rahmen, bestimmt der Zeilen-Parsing-Prozeß die linken, rechten, oberen und unteren Grenzen der durch vertikalen weißen Raum begrenzten Pixelzonen. Die Grenzen werden derart berechnet, daß so wenig Leerraum wie möglich um die Pixelzone verbleibt.

Wenn die resultierende "gerahmte" Pixelzone größer als 64 Pixel (die größte Zeichenbreite, die bei dem beschriebenen Ausführungsbeispiel ohne Reduktion der Auflösung des Zeichens verarbeitet werden kann), oder wenn das Verhältnis der Breite zur Höhe des gerahmten Pixelbereichs größer als 3 : 2 ist, wird angenommen, daß die gerahmte Pixelzone mehr als einen Buchstaben enthält.

In einem solchen Falle kann die gerahmte Pixelzone mit Überhängen versehene, gotische Zeichen enthalten. Derartige Zeichen überlappen einander, obwohl sie sich tatsächlich nicht berühren. In einem solchen Falle kann es vorkommen, daß ein vertikaler Freiraum zwischen den Zeichen nicht existiert. Ein Beispiel kann der Buchstabe "T" gefolgt von "o" sein. Wenn ein "T" entsprechend eng an ein "o" gestellt wird, so verschwindet ein vertikaler Freiraum zwischen diesen beiden Buchstaben.

Ein Auflösungsprozeß wird bei solchen relativ breiten Pixelrahmenzonen angewandt. Der Auflösungsprozeß berechnet den am weitesten links gelegenen freien Weg von der Zeilenober- zur -unterseite. Ein freier Weg wird als Liste von verbundenen vertikalen und horizontalen Segmenten definiert, welche einen freien Weg zwischen zwei Buchstaben verfolgen. Wenn der Auflösungsprozeß durch Auffinden eines freien Weges erfolgreich ist, werden die linken, rechten, oberen und unteren Grenzen für die Pixelrahmenzone links des freien Weges neu berechnet. Wenn der resultierende Rahmen immer noch breiter als 64 Pixel ist oder ein Breiten- zu Höhenverhältnis von mehr als 3 : 2 hat, oder wenn kein freier Weg gefunden wurde, wird der Versuch unternommen, eine Unterstreichung festzustellen und zu entfernen. Wenn dieser Vorgang erfolgreich ist, wird mit Hilfe des Zeilen-Parsing-Prozesses erneut versucht, den vertikalen weißen Raum zu finden.

Nach dem Auffinden eines Pixelrahmens, der nicht zu breit ist, wird dieses Rahmenfeld als isoliertes Zeichen angesehen, Block **232**, und es wird ein Zeichenfenster für die Zeichen-Erkennungsroutinen geschaffen, Block **233**. Bei dem beschriebenen Ausführungsbeispiel der Erfindung ist ein Zeichenfenster ein Pufferbereich, der bis zu 128 Zeilen mal 128 Spalten oder 64 Zeilen mal 192 Spalten enthalten kann. Wenn der Pixelrahmen zu groß ist, um in das Zeichenfenster zu passen, wird er so skaliert, daß er in das Fenster paßt. Der Pixelrahmen wird zeilenweise in das Fenster kopiert. Wenn die gerahmte Pixelzone als Ergebnis eines Auflösungsprozesses abgeleitet wurde, ist die rechte Grenze der Pixelzone als freier Weg beim Auflösungsprozeß definiert. Anderenfalls besteht jede in das Fenster kopierte Zeile aus Bits in entsprechenden Zeilen der Pixelrahmenzone zwischen vertikalen weißen Räumen (d. h. den als isoliert angenommenen Zeichen).

Ein errichtetes Fenster, das eine Spaltenbreite von mehr als 128 Pixeln hat, wird in einen Zurückweisungscache für spätere Weiterbearbeitung gelegt. Anderenfalls werden die Zeichen-Erkennungsroutinen aufgerufen, und das Fenster wird in diese Zeichen-Erkennungsroutinen eingegeben, Block **234**. Wenn ein Zeichen von den Zeichen-Erkennungsroutinen erfolgreich verarbeitet wurde, wird ein Code für seine ermittelte Form in eine als "Fahne" bezeichnete Pufferzone gegeben. Fenster, die von allen Zeichen-Erkennungsroutinen zurückgewiesen wurden, werden zum Zurückweisungscache für spätere Verarbeitung hinzugefügt.

ZEICHENERKENNUNG

Im folgenden wird auf **Fig. 11** Bezug genommen. Danach enthält der Zeichen-Erkennungsprozeß die Schritte der Schablonenanpassung, Block **1101**, gefolgt von der Merkmalsanalyse, Block **1105**, wenn das Zeichen in dem Schablonenanpassungschritt, Block **1101**, nicht erkannt worden ist.

Der Schablonenanpassungsvorgang, Block **1101**, versucht in Fenstern aus dem Zeilen-Parsing-Prozeß weitergeleitete Zeichen mit Schablonen von zuvor bereits identifizierten Zeichen in Übereinstimmung zu bringen. Im Merkmalsanalyseprozeß, Block **1105**, wird versucht, Merkmale der durch Schablonenvergleich nicht identifizierbaren Zeichen zu erken-

nen. Auf der Basis des Erkennen[redacted]her Merkmale werden die Zeichenformen iden[redacted]rt.

Ein Gesichtspunkt der vorliegenden Erfindung besteht darin, daß mit Hilfe des Merkmalsanalysevorgangs erkannte Zeichen als Schablonen zur Erkennung später auftretender Zeichen verwendet werden. Bei dem beschriebenen Ausführungsbeispiel wird ein Schablonen-Cachespeicher für jedes neue Dokument aufgebaut. Der Schablonen-Cachespeicher enthält Zeichen, die innerhalb des Merkmalsanalyseprozesses für das aktuelle Dokument erkannt worden sind. Zeichen in dem Schablonen-Cache werden im Schablonen-Anpaßprozeß benutzt. Durch Errichtung des Schablonen-Cache auf der Basis von bereits im Dokument erkannten Zeichen mit Hilfe des Merkmalserkennungsprozesses macht es die Erfindung möglich, jede durch Merkmalsanalysenroutinen erkennbare Schriftart selbsttätig zu erkennen. Durch Kombinieren von Elementen der Merkmalsanalyse und der Schablonenanpassung bietet die Erfindung die betrieblichen Vorteile eines Schablonenanpaßsystems mit der Allschriftcharakteristik eines Merkmalsanalysesystems.

SCHABLONENANPASSUNG

Der Schablonen-Cache enthält Informationen über jede verfügbare Schablone des aktuellen Dokuments. Für jede Schablone enthält ein Kopffeld Identifizierungsinformationen für diese spezielle Schablone. Das Kopffeld enthält auch Offset-Zeiger bzw. Hinweise auf drei Pixelmuster, die von dem Schablonenanpaßprozeß verwendet werden.

Das erste Pixelmuster ist das Originalmuster des Zeichens, wie es durch den Merkmalsanalyseprozeß erkannt worden ist. Bei dem beschriebenen Ausführungsbeispiel wird das Originalmuster von x Zeilen mal y Spalten als zweidimensionale Matrix gespeichert, wobei die Zeilen Null-unterlegt bis zu einer Wortgrenze sind.

Ein zweites Pixelmuster, bezeichnet als "muß-aus-sein"-Muster, wird aus dem Originalmuster abgeleitet. Das muß-aus-sein-Muster enthält x + 1 Zeilen und y + 1 Spalten, wobei die Zeilen ebenfalls bis zu einer Wortgrenze mit Nullen aufgefüllt sind.

Ein drittes Pixelmuster, genannt "muß-ein-sein"-Muster, wird aus dem Originalmuster abgeleitet und enthält x – 1 Zeilen mal y + 1 Spalten. Das tatsächliche Bild des muß-ein-sein-Musters besetzt nur x – 2 Zeilen mal y – 2 Spalten. Bei dem beschriebenen Ausführungsbeispiel ist jedoch ein Bereich von x – 1 Zeilen mal y + 1 Spalten aus Verarbeitungsgründen reserviert, um sicherzustellen, daß die Matrix ebenso breit wie die muß-aus-sein-Matrix ist.

Wie sich aus der folgenden Beschreibung der Schablonenanpaßmethoden, wie sie bei der Erfindung Verwendung finden, ergibt, werden Zeichen von dem Schablonen-Anpaßprozeß erkannt, wenn die Zeichen innerhalb gewisser vorgegebener Toleranzen der Schablonen liegen. Das Zulassen von Zeichen innerhalb vorgegebener Toleranzen ist wichtig, da zwei Bit-mapped-Bilder desselben Zeichens selten – wenn überhaupt – exakt übereinstimmen. Eine Bilddigitalisierung ist empfindlich gegenüber Unterschieden in der Ausrichtung, und der Digitalisierungsprozeß selbst führt ein Randrauschen ein. Außerdem werden Zeichen oft unterbrochen oder sind auf andere Weise deformiert aufgrund der schlechten Bildqualität, wenn die Zeichen ursprünglich gedruckt wurden, wenn das abgetastete Substrat kopiert wurde oder wenn das Substrat für den Zeichenerkennungsprozeß optisch abgetastet wird. Daher ist ein einfacher Bit-für-Bit-Vergleich bei einem Erkennungsprozeß nicht angemessen. Die muß-ein-sein- und muß-aus-sein-Bildmuster werden bei der Erfindung verwendet, um eine gewisse Differenzgrenze zwischen den Zeichen zu ermöglichen.

Die muß-aus-sein-Matrix enthält ein Pixelbild, welches anzeigt, welche Pixel im Fenster aus (d. h. auf Null gesetzt) sein müssen, um das Fenster mit der Schablone in Übereinstimmung zu bringen. Es sei auf **Fig.** 12(a) Bezug genommen, in der ein Bit-mapped-Bild des Buchstabens e (**1201**) dargestellt ist. Die x's im Buchstaben e bezeichnen Pixel, welche im Originalbild im Schablonen-Cachespeicher eingeschaltet sind.

Bei dem beschriebenen Beispiel zeigt das muß-aus-sein-Bild Pixel an, welche um ein oder mehr Pixel von einem Ein-Pixel des original Bit-mapped-Bildes beabstandet sind. **Fig.** 12(b) zeigt den Buchstaben e (**1202**), bei dem die muß-aus-sein-Pixel als Striche dargestellt sind. Bei dem beschriebenen Ausführungsbeispiel werden die muß-aus-sein-Pixel durch "Verwischen" des Originalpixelbildes berechnet. Das Verwischen wird mit Hilfe logischer ODER-Verknüpfungen in jeder Zeile des Originalpixelbildes vorgenommen. Jede Zeile des Originalpixelbildes wird einer logischen ODER-Verknüpfung mit einer Kopie von sich selbst unterworfen, die um ein Bit nach links verschoben wurde. Das Ergebnis ist logisch ODER-verknüpft mit der ursprünglichen Zeile, verschoben nach rechts um ein Bit. Das Resultat dieses Schrittes wird mit dem Resultat des gleichen Verfahrens an der unmittelbar vorhergehenden Zeile oder darüber logisch ODER-verknüpft. Das Resultat dieses Schritts wird logisch ODER-verknüpft mit der ähnlich verarbeiteten Zeile unmittelbar vor oder unterhalb der aktuellen Zeile. Das Ergebnis dieser Operation ist ein Bild des ursprünglichen Zeichens, bei dem jedes Pixel des ursprünglichen Bit-mapped-Bildes von 8 Ein-Pixeln umgeben ist, von dem Pixel oberhalb, unterhalb, rechts, links und den 4 Pixeln, die um 45°, 135°, 225° und 315° vom ursprünglichen Pixel verdreht sind. Dadurch wird das ursprüngliche Zeichenbild mit einer Ein-Pixel-Schicht quasi umgürtet. Das Komplement des sich ergebenden Bildes wird als muß-aus-sein-Muster erhalten.

Das muß-ein-sein-Bild ist eine Zeichenabbildung mit Bits, die getastet sein müssen für eine Übereinstimmung. In **Fig.** 12(b) ist die Zeichenabbildung des Zeichens e **1202** mit Pluszeichen dargestellt, die bei einer Übereinstimmung eine 1 bedeutende Pixel zeigen. Zum Berechnen eines muß-ein-sein-Bildes wird jede Zeile des originalen Pixelbildes logisch UND-verknüpft mit einer um ein Bit nach links verschobenen Kopie der Zeile. Das Ergebnis wird mit einer Kopie der um ein Bit nach rechts verschobenen Zeile logisch UND-verknüpft. Deren Ergebnis wird logisch UND-verknüpft mit einem ähnlich behandelten Bild der Zeile unmittelbar über der aktuellen Zeile. Das Ergebnis der Operation wird dann logisch UND-verknüpft mit der ähnlich verarbeiteten Zeile unmittelbar unterhalb der aktuellen Zeile. Diese logische UND-Operation mit Zeilen oberhalb und unterhalb der aktuellen Zeile findet statt, nachdem die oberhalb und unterhalb der aktuellen Zeile gelegenen Zeilen zuvor wie in den ersten zwei Schritten dieses Verfahrens beschrieben mit ihren eigenen Bildern logisch UND-verknüpft worden sind. Durch dieses Verfahren wird eine Abbildung des Originalzeichens erzeugt, bei dem nur diejenigen Pixel eingeschaltet bleiben, die auf allen ihren acht Seiten umgeben sind. Dadurch entsteht im Ergebnis ein Bild, das um eine Pixelschicht dünner als das Originalbild ist.

Die Verwendung der muß-ein-sein- und muß-aus-sein-Matrix zum Vergleich der Eingangszeichen mit Schablonen ermöglicht Toleranzen bei der Durchführung der Anpassung. Obwohl das beschriebene Ausführungsbeispiel eine Ein-Pi-

xel-Toleranz beim Prüfen [redacted] Übereinstimmungen zuläßt, ist es für den Fachmann klar, daß andere Ausführungen zu anderen Toleranzen führen. Ein alternatives Ausführungsbeispiel, das weniger enge Toleranzen ermöglicht, kann zu höheren Anpaßgeschwindigkeiten und daher zu entsprechend rascherer Verarbeitung führen. Ein solches Ausführungsbeispiel erhöht jedoch die Fehlerrate der Identifizierung von Zeichen aufgrund der weniger strengen Toleranzanforderungen.

Im folgenden wird auf **Fig.** 11(b) Bezug genommen. Bei jedem Empfang eines neuen Fensters mit nicht-identifizierter Pixelinformation aus der Zeilen-Parsingroutine werden muß-ein-sein- und muß-aus-sein-Bilder für das nicht-identifizierte Bild unter Verwendung des oben beschriebenen Verfahrens erzeugt, Block **1120**. Das nicht-identifizierte Bild im Fenster wird dann mit den Zeichen im Schablonencache verglichen. Die Schablonen werden zu Vergleichszwecken in einer zuletzt angepaßten Reihenfolge geordnet. Bei jeder Übereinstimmung mit einer Schablone wird ein im Schablonen-Kopfetikett gespeicherter Zählwert inkrementiert.

Wenn eine Schablone als Ergebnis der Erkennung durch die Merkmalsanalysenroutinen das erste Mal hergestellt wird, wird der Schablonenübereinstimmungszähler auf 0 gesetzt. Bei dem beschriebenen Beispiel werden neue Schablonen (d. h. Schablonen mit einem Übereinstimmungszählwert von 0) bei Beginn der Schablonenschlange eingesetzt. Wenn ein nicht-identifiziertes Bild von den Schablonen-Anpaßroutinen verarbeitet wird und mit einer besonderen Schablone übereinstimmt, wird der zur besonderen Schablone gehörige Übereinstimmungszähler geprüft, um festzustellen, ob der Anpassungszählwert 0 ist. Wenn der Zählwert 0 ist, prüft das beschriebene Ausführungsbeispiel der Erfindung das Bild im Zeichenfenster unter Verwendung der Merkmalsanalysenroutinen (weiter unten beschrieben), um eine Bestätigung dafür zu erhalten, daß das Bild im Zeichenfenster das gleiche Zeichen wie das von der Schablone identifizierte Zeichen ist. Wenn die Merkmalsanalysenroutinen die Schablone bestätigen und das Bild im Zeichenfenster das gleiche Zeichen ist, wird der Anpaßzählwert inkrementiert. Anderenfalls wird im Verfahren unterstellt, daß die Schablone zu unzuverlässigen Ergebnissen führt, und die Schablone wird aus der weiteren Verarbeitung eliminiert. Die Erkennung des Bildes im Zeichenfenster wird dann dadurch fortgesetzt, daß man das Bild im Zeichenfenster mit anderen Schablonen im Schablonencache in Übereinstimmung zu bringen sucht.

Der erste Schritt beim Anpassen eines Bildes in einem Fenster mit der Schablone besteht darin, daß man muß-ein-sein- und muß-aus-sein-Matritzen von dem nicht-identifizierten Bild konstruiert, Block **1120**. Als nächstes wird eine Dimensionsprüfung durchgeführt, Block **1121**. Bilder, die sich von einer Schablone bezüglich ihrer Höhe oder Breite um mehr als ein Pixel unterscheiden, können zu der Schablone nicht passen, Zweig **1122**. Wenn die Dimensionsprüfung durchlaufen ist, Zweig **1123**, wird die muß-ein-sein-Matrix für das nicht-identifizierte Bild im Fenster mit dem original Bit-mapped-Bild der Schablone verglichen. Wenn alle Pixel in der muß-ein-sein-Matrix für das nicht-identifizierte Bild in der Originalschablone ein-getastet sind, Zweig **1124**, wird ein zweiter Test durchgeführt.

Der zweite Test bestimmt, ob alle Pixel in der muß-ein-sein-Matrix für die Schablone in dem nicht-identifizierten Bit-mapped-Bild im Fenster eingeschaltet sind. Wenn alle derartigen Bits eingeschaltet sind, Zweig **1125**, wird das Original Bit-mapped-Bild der Schablone mit der muß-aus-sein-Matrix für das Bild im Fenster verglichen. Alle in der muß-aus-sein-Matrix für das Bild im Fenster angegebenen Pixel müssen in der ursprünglichen Schablone als Voraussetzung einer Anpassung oder Übereinstimmung ausgeschaltet sein. Wenn dieser Test durchgeführt ist, Zweig **1126**, wird das Bit-mapped-Bild im Fenster mit der muß-aus-sein-Matrix für die Schablone verglichen. Wenn alle von der muß-aus-sein-Matrix der Schablone angezeigten Pixel im Bit-mapped-Bild des Fensters ausgeschaltet sind, wird die Schablone als übereinstimmend oder angepaßt bewertet und der Zweig **1127** eingeschlagen.

Wie oben beschrieben, wird das Bild im Zeichenfenster auch mit den Merkmalsanalyseroutinen zur Bestätigung der Identifizierung analysiert, wenn die Schablone eine Übereinstimmungszählung von 0 hat. Anderenfalls wird der Identifizierungscode für die erkannte Form in die Fahne für eine spätere Verarbeitung durch die Kontext-Analysenroutinen eingegeben.

Wenn einer der Zweige **1122, 1128, 1129, 1130** oder **1131** eingeschlagen wird, weil der entsprechende oben beschriebene Test nicht passiert wurde, und wenn mehrere Schablonen im Schablonenspeicher vorhanden sind, Zweig **1132**, wird die gleiche Folge von Prüfungen oder Tests mit jeder nachfolgenden Schablone im Schablonencache solange wiederholt, bis eine Anpassung oder Übereinstimmung auftritt oder der Cache erschöpft ist.

Wenn es keine weiteren Schablonen im Schablonencache gibt, Zweig **1133**, gibt es keine Übereinstimmung zwischen den aktuellen Schablonen und dem nicht-identifizierten Bild. Dies ist immer dann der Fall, wenn sich das nicht-identifizierte Bild nach Schriftart, Größe oder Ausrichtung von allen im Schablonen-Cachespeicher enthaltenen Zeichen unterscheidet. Die fehlende Übereinstimmungsbedingung kann auch das Ergebnis von Zeichen sein, die trotz gleicher Schriftart und -größe nicht eng genug übereinstimmen, um innerhalb der "Randrausch"toleranzen der Schablonenanpassungsroutinen zu liegen.

In jedem Falle werden die Merkmalsanalyseroutinen herangezogen, die das Bild im Fenster als Eingabe benutzen, Block **1105**, wenn das Bild zuvor nicht erkannt werden konnte, Zweig **1104** in **Fig.** 11(a).

MERKMALSANALYSE

Bei dem beschriebenen Ausführungsbeispiel wird eine Vielzahl von Routinen zum Analysieren der Merkmale der den Merkmalsanalyseprozeß durchlaufenden Bilder verwendet, wobei die Kategorie der Form eines nicht-identifizierten Bildes im Zeichenfenster bestimmt wird. Unter den verschiedenen Routinen ist eine Routine für jede besondere Spezies im normalen Zeichensatz. Jede dieser einzelnen Routinen kann ein Bild in einem Zeichenfenster analysieren und als Ausgabe eine Anzeige dafür zur Verfügung stellen, ob das Bild zu der allgemeinen Formkategorie zählt, die durch diese Routine zu unterscheiden ist. Die Zeichenerkennungsroutinen werden aktiviert, wenn eine der Routinen mit einer positiven Anzeige dafür antwortet, daß das Bild im Zeichenfenster die der speziellen Routine entsprechende Form hat. Wenn keine der Merkmalsanalysenroutinen positiv antworten, bleibt die Form des Bildes im Zeichenfenster unidentifiziert. In einem solchen Falle wird eine Weiterverarbeitung durchgeführt, um die Form des Bildes im Zeichenfenster zu identifizieren.

Jede der Routinen wird als "isit" bezeichnet. Der Name "isit" ist für die Beschreibung der Routinen des Ausführungs-

beispiels zweckmäßig, da die Ro[redacted] bestimmen, ob ein Zeichen im Zeichenfenster [redacted]ezielles Zeichen ist (z. B. "ist es" (isit) ein a). Bei dem beschriebenen Ausführungsbeispiel existieren isits für Buchstaben, Zahlen und spezielle Symbole, wie Kommas, Anführungsstriche, Semikolons usw.. Es ist für den Fachmann klar, daß das Verfahren der Verwendung von isits zur Bestimmung der Zugehörigkeit eines Bildes in einem Zeichenfenster zu einem speziellen Zeichen auf zahlreiche alphabetische Zeichengruppen anwendbar ist. Beispielsweise können isits für kyrillische Zeichensätze, die Zeichensätze in slawischen Sprachen, oder für andere Zeichensätze, z. B. die Zeichensätze für Hebräisch oder Arabisch, implementiert werden.

Bei dem beschriebenen Ausführungsbeispiel unterscheiden isits Zeichen aufgrund ihrer Form. Daher werden Zeichen, welche die gleiche Togographie haben, von einem einzigen isit erkannt. Der Kleinbuchstabe "p" und der Großbuchstabe "P" werden von demselben isit erkannt. Die Buchstaben "u" und "U", "s" und "S", "o" und "O" sowie "0" usw. sind weitere Beispiele für Zeichen, die mit der gleichen oder ähnlichen Topographie versehen sind und daher von denselben isits erkannt werden. Für jede Form oder Topographie wurden Formcharakteristiken gewählt und experimentell gemessen, so daß ein besonderes isit die Form seines Zeichens aus der Form anderer Zeichen über einen weiten Bereich von Schriftarten unterscheiden kann.

Bei dem beschriebenen Ausführungsbeispiel liefert ein isit als Ausgang entweder einen ASCII-Code für ein spezielles Zeichen oder einen Code, der anzeigt, daß das Zeichen als zugehörig zu einer speziellen Zeichenklasse erkannt worden ist, oder einen Zurückweisungscode, der anzeigt, daß das Zeichen nicht erkannt worden ist. Die Ausgabe des ASCII-Code für ein spezielles Zeichen zeigt an, daß die Zeichenidentifizierung durch die isit-Routine unzweideutig ist. Der zurückgeführte ASCII-Code ist der Standard-ASCII-Code für das spezielle Zeichen. Ein Code, der angibt, daß das Zeichen zu einer besonderen Zeichenklasse gehört, beschränkt jede nachfolgende Merkmalsanalyse auf einen speziellen Satz oder eine spezielle Gruppe von isits.

Im folgenden wird auf **Fig.** 13 Bezug genommen, in der ein Ablaufdiagramm des beschriebenen Merkmalsanalyseprozesses dargestellt ist. Für jedes dem Merkmalsanalyseprozeß als Eingabe zugeführte Bild werden statistische Daten aus einem Horizontalfenster und einem Vertikalfenster von den isits benutzt. Das Horizontalfenster ist das ursprüngliche Zeichenfenster mit der Bit-map-Abbildung des Zeichens. In **Fig.** 14(a) ist das Zeichen "b" beispielsweise im horizontalen Zeichenfenster **1401** gezeigt. Ein Vertikalfenster wird aus dem Horizontalfenster **1401** abgeleitet, Block **1301**. Das Vertikalfenster kann als ein auf der Seite liegendes Bild des Horizontalfensters **1401** angesehen werden, dessen Zeilen jeweils eine Positionsumkehr erfahren haben. So zeigt beispielsweise **Fig.** 14(b) ein Vertikalfenster für das Bild des Zeichens "b" im Vertikalfenster **1410**.

Statistische Informationen werden durch Prüfen der Bit-map-Bilder im Horizontalfenster **1401** und im Vertikalfenster **1410** erzeugt. Die statistischen Daten enthalten Profildaten, Polygondarstellungen des Zeichens, Phasenänderungsinformationen und Zählungen der Anzahl von Pixeln in jeder Zeile des Zeichens.

Die bei den beschriebenen erfindungsgemäßen Verfahren angewandten Polygon-Anpaßalgorithmen dämpfen die Effekte des Rauschens in den zu identifizierenden Bildern und reduzieren beträchtlich das von den Merkmalsrichtungsroutinen zu verarbeitende Datenvolumen. Es wurde außerdem bestimmt, daß Polygondarstellungen von Zeichenbildern über einen weiten Bereich von Zeichengrößen konsistent sind, z. B. der Buchstabe "i" erzeugt im wesentlichen die gleichen Polygondarstellungen in einem weiten Bereich von Schrifttypen.

Für jede Ansicht eines Zeichenfensters werden Profildaten und vier Polygone abgeleitet. Die Ansichten eines Zeichens enthalten die linken und rechten Seiten **1403**, **1404**, **1413** und **1414** der horizontalen und vertikalen Fenster. Die Profildaten enthalten ein Feld mit einem Element für jede Zeile des Fensters. Jedes Element hält einen Wert, gemessen in Spalten, der den Abstand des Rahmenrandes vom ersten "Ein-Pixel" in dieser Zeile darstellt. Es wird beispielsweise auf **Fig.** 14(a) Bezug genommen. Das erste Ein-Pixel in jeder Zeile ist bei der Ansicht **1403** das erste Ein-Pixel im Anstrich des Buchstabens b. In der Ansicht **1404** ist das erste Ein-Pixel für die obere Hälfte des Buchstabens b ebenfalls das erste Ein-Pixel im Anstrich des Buchstabens b. Für die untere Hälfte der Ansicht **1404** ist das erste Ein-Pixel für jede Zeile das am äußeren Rand der Schleife **1405** gelegene Pixel des Buchstabens b.

Ein erstes Polygon wird für jede Ansicht gebildet. Das Polygon enthält mehrere Zeilensegmente, die einem Profil des Zeichens in Richtung der jeweiligen Ansicht angepaßt ist. Die Zeilensegmente des Polygons liegen innerhalb vorgegebener Toleranzwerte für das Profil des Zeichens. Beispielsweise ist in **Fig.** 14(c) ein Profil **1420** des Buchstabens b der Ansicht **1404** in **Fig.** 14(a) gezeigt. Die Punkte **1421** bis **1429** sind die Polygonpunkte welche dieses Profil beschreiben. Bei dem beschriebenen Beispiel gibt es maximal 16 Punkte im Polygon zur Beschreibung des speziellen Profils, und für jedes Segment werden Steigung und Unterschied der Steigung gegenüber einem früheren Segment berechnet und aufrechterhalten. Es ist für den Fachmann klar, daß eine große Anzahl von Punkten, verbunden mit einer entsprechenden Zunahme an Bearbeitungs- und Speichermitteln zur Beschreibung eines Polygons verwendet werden kann.

Die Polygon-Anpaßalgorithmen bestimmen die Punkte, wie die Punkte **1421** bis **1429** auf dem Profilbild **1420**. Der erste Schritt beim Polygon-Anpaßprozeß ist die Zuordnung von Polygonpunkten **1421** und **1429** an jedem Ende des Polygons. Eine Rekursivroutine wird aufgerufen, in der die Endpunkte eines Liniensegments, beispielsweise die Punkte **1421** und **1429**, das Profilbild **1420** und ein Toleranzwert als Eingaben verwendet werden. Der Toleranzwert bestimmt die "Enge" der Anpassung des Polygons an das Profilbild **1420**.

Bei dem beschriebenen Ausführungsbeispiel wird die Toleranz (t) in 128steln eines Pixels gemessen und auf der Basis der langen und kurzen Abmessungen eines Fensters nach der folgenden Formel berechnet:

Toleranz (t) = (13/4)x + 64, wenn x < 28; und (t) = 5x, wenn x ≥ 28;
wobei x = (3* (Länge der langen Seite) + (Länge der kurzen Seite))/4.

Der Polygon-Anpaßalgorithmus zieht effektiv eine Linie zwischen die Endpunkte **1421** und **1429** und lokalisiert die am weitesten oberhalb und unterhalb der Linie (Punkte **1422** und **1426**) gelegenen Punkte. Wenn jeder Extrempunkt jenseits der zulässigen Toleranz liegt, wird er in das Polygon einbezogen, wodurch das ursprüngliche Zeilensegment in Untersegmente unterbrochen wird. Der Algorithmus wird durch rekursives Anwenden des gleichen Verfahrens auf jedes

Untersegment solange fortgesetzt, bis kein Datenpunkt weiter als die zulässige Toleranz von dem nächsten Polygonsegment entfernt liegt. Bei dem aktuellen Beispiel liegen beide Extrempunkte (**1422** und **1426**) außerhalb der zulässigen Toleranz, so daß das ursprüngliche Liniensegment in drei Untersegmente **1421** bis **1427**, **1422** bis **1426** und **1426** bis **1249** unterbrochen wird. Der Algorithmus wird durch Zeichnen des Liniensegmentes zwischen Punkten **1421** und **1422** fortgesetzt. Dieses Liniensegment hat keine Punkte außerhalb des Toleranzbereichs ober- und unterhalb, so daß es nicht weiter unterteilt ist. Der Algorithmus zeichnet dann die Linie zwischen den Punkten **1422** und **1426** und lokalisiert Punkte, die am weitesten oberhalb und unterhalb der Linie liegen. In diesem Falle wird der Punkt **1425** als der am weitesten über dem Toleranzwert gelegene Punkt bestimmt, während kein Punkt unterhalb liegt. Dies schafft zwei neue Untersegmente, nämlich **1422** bis **1425** und **1425** bis **1426**. Diese werden rekursiv weiterentwickelt, bevor in dem Prozeß das letzte Untersegment geprüft wird, das sich aus der ersten Iteration des Prozesses ergeben hat. Der Prozeß zieht eventuell eine Linie zwischen Punkt **1426** und **1429** und stellt fest, daß Punkt **1428** am weitesten oberhalb der Linie liegt und keine Punkte jenseits des Toleranzwerts unterhalb liegen. Die sich ergebenden Untersegmente werden in ähnlicher Weise weiterentwickelt durch rekursive Anwendung des gleichen Verfahrens.

Der Prozeß der iterativen Erzeugung von Untersegmenten durch Bestimmung von Punkten, welche maximal oberhalb und unterhalb der Toleranzgrenzen existierender Segmente liegen, wird solange fortgesetzt, bis kein Ursprungsdatenpunkt mehr weiter als der Toleranzwert von dem nächsten Polygonsegment des Profils **1420** entfernt liegt.

Ein zweites Polygon weist Liniensegmente auf, welche Punkte verbinden, die einer Darstellung einer Profilansicht eines als Schattenprofil bezeichneten Zeichens angepaßt sind. Das Schattenprofil wird von einem Profil der Polygonansicht dadurch abgeleitet, daß das Polygon von der Bodenzeile zur Kopfzeile durchlaufen und der minimale X-Wert beim Durchlaufen des Polygons vom aktuellen Außenpunkt auf dem Polygon subtrahiert wird. Derselbe Vorgang wird danach von der Kopfzeile zur Bodenzeile des Polygons wiederholt. Der Effekt ist, daß die restlichen nicht-Null-X-Punkte im Profil Zonen darstellen, welche effektiv abgeschaltet wären, wenn der Buchstabe von Licht auf beiden Seiten beleuchtet wäre. Das Schattenpolygon dient zur Bestimmung und Analyse von Öffnungen in das Innere eines Zeichens.

Ein drittes Polygon enthält Profilpunkte einer Ansicht eines Zeichens, welche durch Subtraktion des Schattenprofils vom ursprünglichen Profil gebildet werden. Dieses Polygon wird als gefülltes Polygon bezeichnet. Sowohl das Schattenpolygon als auch das gefüllte Polygon haben die gleiche Anzahl von Punkten und die gleichen Y-Koordinaten wie das Originalpolygon.

Schließlich wird ein viertes Polygon gebildet, das eine losere Anpassung an das Originalpolygon als das ursprüngliche Profil hat. Dies geschieht durch Entwicklung eines Polygons aus dem Ursprungspolygon unter Verwendung weniger Punkte (d. h. unter Verwendung eines weiteren Toleranzwertes).

Daten werden sowohl für das horizontale als auch das vertikale Fenster entwickelt, welche die Zahl von Phasenwechseln von weiß nach schwarz in jeder Zeile des entsprechenden Fensters zählen. Beispielsweise besteht im Horizontalfenster der Buchstabe "I" generell aus Zeilen mit einer horizontalen Phasenänderung von weiß nach schwarz und der Buchstabe "H" besteht aus Zeilen mit zwei Phasenänderungen von weiß nach schwarz. Die Phasenänderungsinformationen dienen zur Erzeugung von durch die isit-Routinen verwendeten Statistiken.

Eine erste Statistik wird entwickelt, welche den Prozentsatz von Zeilen mit einem besonderen Phasenänderungswert bezeichnet. So kann beispielsweise für den Buchstaben "I" der Wert 1 (Auftreten einer Phasenänderung von weiß nach schwarz in der Zeile) nahezu bei 100% der Zeit auftreten. Bei dem Buchstaben "H" kann der Wert 1 über 5% der Zeit und der Wert 2 über etwa 85% der Zeit auftreten. Bei dem Buchstaben "d" können in dem Horizontalfenster 0-Phasenänderungszählungen des Werts 0 auftreten, was bedeutet, daß jede Zeile wenigstens ein Pixel eingeschaltet hat. Es können angenähert 55% der Zeilen eine Phasenänderung erfahren, und diese Zeilen enthalten die obere Hälfte des Buchstabens "d". Die restlichen ca. 45% der Zeilen haben zwei Phasenänderungen, und diese Zeilen umfassen die Schleife des Buchstabens "d". Es gibt keine Zeilen mit drei oder vier Phasenänderungen.

Bei dem beschriebenen Ausführungsbeispiel werden Zeilen mit mehr als vier Phasenänderungen wie Zeilen mit genau vier Phasenänderungen gezählt, und zwar zum Zwecke der Berechnung dieser Prozentwerte. Die Grenze von vier Phasenänderungen wurde von der Anmelderin aus experimentellen Daten und Beobachtungen entwickelt. Typischerweise treten mehr als vier Phasenänderungen bei starken Rauscheinflüssen im Bit-mapped- Bild auf.

Eine zweite Statistik wird entwickelt, die angibt, wo die Mitte einer besonderen Phasenänderungszählung im Zeichenbild auftritt. Bei dem Buchstaben "I" ist der "wo"-Wert für die Phasenänderungszählung von 1 nahe bei 50. Dies zeigt die Mittelstellung einer Linie mit einer Phasenänderungszählung von 1 ungefähr in der Zeichenmitte an. Bei dem Buchstaben "d" ist der wo-Wert für die Phasenänderungszählung des Werts von 1 etwa bei 20, was angibt, daß der Mittelwert der Zeilenzahlen mit einer einzigen Phasenänderung etwa bei 20% des Weges abwärts des Zeichens liegt. Der wo-Wert für eine Phasenänderungszählung von 2 kann etwa 75 sein, was anzeigt, daß der Mittelwert der Zeilenzahlen mit zwei Phasenänderungen etwa drei Viertel des Weges abwärts des Buchstabens liegt. Dies liegt daran, daß die Schleife des Kleinbuchstabens "d" in der unteren Hälfte des Zeichens liegt.

Es wird auch ein Feld sowohl für das horizontale als auch das vertikale Fenster entwickelt, das ein Element für jede Zeile des Fensters mit einem Wert für die Anzahl von "Ein"-Pixeln in dieser Zeile enthält. Die Polygone werden auch an diese Felder angepaßt.

Die isits dienen zur individuellen Analyse des Zeichens, Block **1304**, bis das Zeichen erkannt ist. Eine besondere isit-Routine kann von irgendeiner aus einer Anzahl von Methoden Gebrauch machen, um zu identifizieren, ob ein Zeichen gleich dem durch das besondere isit zu identifizierenden Zeichen ist. In einigen Fällen, in denen sich der Stil eines Zeichens von Schriftart zu Schriftart beträchtlich ändert, können mehrere unterschiedliche Methoden zur Erkennung des Zeichens verwendet werden. Typischerweise analysieren die isit-Routinen die Form der einzelnen Zeichenansichten durch Verwendung der oben beschriebenen Polygone und durch Verwendung der oben angegebenen statistischen Informationen. Die Verwendung von Polygonen zur Approximation von Zeichen erlaubt es einer isit-Routine in der Regel, kleinere Störungen und Unregelmäßigkeiten des Zeichens zu ignorieren.

Eine erste Methode, die von isits zur Bestimmung eines Bildes des Zeichens benutzt wird, besteht in der Prüfung der statistischen Informationen aus den Phasenänderungsdaten. So kann beispielsweise ein spezielles isit ein Zeichen zu-

rückweisen, in welchem der Prozentsatz von Zeilen mit zwei Änderungen von weiß nach schwarz 10% übersteigt. Ein Beispiel eines solchen isit ist ein isit zur Erkennung des Zeichens "I". Generell macht es diese Methode der Untersuchung statistischer Informationen für die meisten isits möglich, 70 bis 85% aller ihnen als Eingaben zugeführten Bilder mit einem Minimum an Verarbeitungsmitteln zu eliminieren.

Ein anderes Verfahren, mit dem bestimmt werden kann, ob ein spezielles Zeichen ein von einem speziellen isit zu erkennendes Zeichen ist, ist die Prüfung von Spitzen in den Polygonen. So ist beispielsweise der Großbuchstabe "F" dadurch gekennzeichnet, daß er zwei extreme Spitzen bei Betrachtung von seiner rechten Horizontalseite aus aufweist. isits können Zeichen sowohl aufgrund der Anzahl von Spitzen als auch deren Relativlage identifizieren. So muß beispielsweise ein isit zur Erkennung des Buchstabens "F" den Buchstaben "c" zurückweisen, da die Anzahl und relative Lage der Spitzen in einer seiner Ansichten oder in einer seiner Populationszählfelder nicht richtig ist.

Einige Zeichen sind gekennzeichnet durch einen "Balken" oder eine "Rampe" (das ist die linke Ansicht eines B, b, h, k, A, D, E usw.). Ein isit zur Erkennung solcher Zeichen kann ein Zeichen auf das längste Einzelsegment im Zeichen untersuchen und nach Charakteristiken, wie dem Prozentsatz der Länge des längsten Segments zur Länge des Zeichens als Ganzem und der Steigung des Segments suchen.

Eine andere Methode, die von isits zur Identifizierung von Zeichen benutzt wird, ist die Identifizierung von Schleifen in den Zeichen. Eine Schleife kann als Linie mit primär zwei Änderungen von weiß nach schwarz über jede Zeile der Schleife identifiziert werden. Die isits identifizieren die Steilheit der Kurve der Schleife, die relative Symmetrie der Kurve und Informationen über Ecken der Schleife. So kann beispielsweise die Zahl "8" häufig von dem Buchstaben "B" aufgrund der Unterschiede in den Ecken unterschieden werden.

Wie oben gesagt, schickt ein isit nach Beendigung einer Analyse entweder den ASCII-Code oder den Formcode für das spezielle Bild oder die Information unter Angabe des Zurückweisungsgrundes zurück. Wenn das Bild erkannt wurde, Zweig **1306**, wird eine Schablone in der oben beschriebenen Weise in Verbindung mit der Schablonenanpassung, Block **1307**, gebildet. Der Identifizierungscode wird dann auf die Zeichenfahne bewegt.

Anderenfalls wird das nächste isit auf der Basis von Parametern, wie die Häufigkeit des durch das isit dargestellten Zeichens und aufgrund von aus vorhergehenden isits gewonnenen Informationen über die Gründe für die Zurückweisung des Zeichens, Zweig **1310**, ausgewählt. Wenn keine isits mehr vorhanden sind, Zweig **1311**, wird das Bild im Zeichenfenster in den Zurückweisungscache bewegt. Anderenfalls, Zweig **1312**, wird das Zeichen vom nächsten isit analysiert, Block **1304**.

SEITENANPASSUNG

In vielen Fällen kann ein Zeichen weder durch die Schablonenanpassung noch durch die Merkmalsanalysenroutinen erkannt werden, da das Zeichen an einem benachbarten Zeichen "klebt". Dies geschieht beispielsweise dann, wenn Zeichen aufgrund der Qualität des Druckprozesses zu eng zusammenhängen. Als typisches Beispiel können zusammengeklebte Zeichen auftreten, wenn die Druckfarbe, Tusche oder Tinte in einem aus Proportionaltext zusammengesetzten Dokument zerfließt.

Es wird ein Verfahren, genannt "Seitenanpassung", zum Identifizieren derartiger zusammenfließender Zeichen beschrieben. Beim Analysieren eines Zeichenfensters unter Verwendung der Seitenanpassung werden Schablonen effektiv auf die Seiten des Zeichenfensters gelegt, um festzustellen, ob eine Übereinstimmung mit dieser Schablone auftreten würde, wenn alle Pixel im "Schatten" dieser Schablone ignoriert würden. Die Seitenanpassungsmethoden machen von den gleichen Schablonenvergleichsalgorithmen Gebrauch, die oben unter Bezugnahme auf die Schablonenanpassung diskutiert wurden; jedoch ignorieren die Seitenanpassungsmethoden die Bilder auf den rechten oder linken Seiten.

Wenn beispielsweise ein Zeichenfenster das Zeichen "ite" enthält, wobei alle Buchstaben zusammengeklebt sind, sucht der Seitenanpaßprozeß Schablonen auf die linken und rechten Seiten des Zeichenfensters zu legen. Jede dieser Schablonen wird dabei mit der linken Seite des Zeichenfensters ausgerichtet, um eine Übereinstimmung auf der linken Seite zu suchen, während Pixel rechts von den Ein-Pixeln in der Schablone ignoriert werden. Bei dem aktuellen Beispiel würde ein Auflegen einer Schablone mit dem Zeichen "i" auf die linke Seite des Zeichenfensters eine Übereinstimmung ergeben. Ist dies der Fall, so wird der ASCII-Code für ein "i" in der Fahne registriert, und der Seitenanpaßprozeß entfernt die den Buchstaben "i" darstellenden Pixel aus dem Fenster. Das Verfahren wird danach fortgesetzt, indem eine Seitenanpassung an den restlichen Pixeln im Fenster versucht wird. In diesem Falle würde eine Übereinstimmung auftreten, wenn eine Schablone für das Zeichen "t" verwendet würde.

Die Seitenanpassungsmethode kann sowohl für die rechte als auch für die linke Seite eines Zeichenfensters benutzt werden und wird angewandt auf alle möglichen vertikalen Ausrichtungen. Wenn die Seitenanpassung von der linken Seite des Zeichenfensters unerkannte Zeichen übrig läßt, wird die Seitenanpassung von der rechten Seite aus versucht.

Da einige Buchstabenformen Untergruppen von anderen sind (z. B. r-n-m oder c-o-d), werden die Schablonen vor ihrer Verwendung bei der Seitenanpassung nach der Größe geordnet. Einige Schablonen (z. B. ein Punkt) sind vom Seitenanpaßprozeß deshalb ausgeschlossen, da derartige Schablonen Übereinstimmungen mit praktisch allen Bildern ergeben.

Bei dem beschriebenen Beispiel wird die Seitenanpassung an Zeichenfenstern im Zurückweisungscache nach Beendigung der Verarbeitung aller Zeichen im Dokument versucht. Dies ermöglicht die Erzeugung einer relativ großen Bibliothek von Schablonen und erhöht die Chancen einer erfolgreichen Identifizierung von Buchstaben mit Hilfe der Seitenanpaßmethode.

KONTEXTANALYSE

Mit dem Zeichenerkennungsverfahren werden Zeichen nach ihrer Form identifiziert. Die Form des Zeichens allein ist jedoch in einigen Fällen nicht eindeutig bestimmend für das Zeichen selbst. Beispielsweise läßt sich der Kleinbuchstabe "s" unter Umständen nicht von dem Großbuchstaben "S" unterscheiden. Als anderes Beispiel ist ein Apostroph aufgrund seiner Form allein nicht von einem Komma unterscheidbar. Die Kontextanalysenroutine akzeptiert als Eingabe und ver-

wendet als Ausgabe die [redacted] der Zeichencodes, wie sie von den Zeichenerkennungsroutinen erzeugt wird. Eine Kontextanalyse wird an einer Seitenzeile oder -linie gleichzeitig durchgeführt, und zwar in dem Versuch, Unsicherheiten zu klären. Im folgenden wird auf **Fig. 15** Bezug genommen, in der das Verfahren nach einem bevorzugten Beispiel der Erfindung dargestellt ist.

Das beschriebene Beispiel der Erfindung enthält eine Datenbank mit charakteristischen Attributen der verschiedenen Zeichen. Diese Attribute können Informationen darüber beinhalten, ob das Zeichen typischerweise vollständig über der Grundlinie der Zeichenzeile liegt oder ob eine Unterlänge oder ein anderer Teil des Zeichens sich typischerweise unter die Grundlinie erstreckt. Die Datenbank enthält auch Informationen über die relative Größe der Zeichen, die normalerweise zweifelhaft ist, wenn nach der Form allein identifiziert wird. So kann die Datenbank beispielsweise Informationen zur Unterscheidung zwischen Groß- und Kleinbuchstaben "S" und "s" auf der Basis der erwarteten Relativgröße enthalten.

Jede Zeile der Seite wird aus der Fahne in einen Puffer zur Vorbereitung der Zeile für die Weiterverarbeitung kopiert, Block **1501**. Während dieses Kopiervorgangs einer Zeile in den Puffer werden den Zeichen aus der Zeichenattribut-Datenbank Werte zugeordnet, wie Informationen darüber, ob die Zeichen über die Basislinie hinausgehen und ob die relative Größe des Zeichens einen Groß- oder Kleinbuchstaben charakterisiert. Der Abstand zwischen Worten wird ebenfalls an dieser Stelle dadurch bestimmt, daß ein Histogramm der Abstände zwischen Buchstaben konstruiert und analysiert wird. Wenn Mehrdeutigkeiten für ein Zeichen geklärt werden, macht die Erfindung von bei der Klärung dieser Mehrdeutigkeiten gewonnenen Informationen Gebrauch, um Mehrdeutigkeiten der benachbarten Zeichen zu lösen.

Die Basislinie wird auch bezüglich der Schräglage eingestellt. Bei dem beschriebenen Ausführungsbeispiel kann die Schräglage dadurch eingestellt werden, daß die erwartete Basislinie für jeden Einzelbuchstaben geprüft und die Basislinie für die Gesamtzeile aufgrund dieser Information eingestellt wird. Wenn die Werte für die Basislinie aber von Zeichen zu Zeichen oder von Wort zu Wort wesentlich voneinander abweichen, wird ein Zeichen nahe jedes Zeilenendes gesucht, das eine mit ausreichender Zuverlässigkeit auf der Basislinie befindliche Position hat (z. B. der Buchstabe "B" sitzt auf der Basislinie, während die Lage des Buchstabens "Q" nicht als ausreichend zuverlässig auf der Basislinie gelegen angesehen werden kann, da bei einigen Schrifttypen ein Fuß oder Abstrich über die Basislinie hinausgehen kann). Danach wird eine eingestellte Basislinie dadurch bestimmt, daß man eine den Boden dieser beiden Zeichen nahe jedes Zeilenendes verbindende Gerade zeichnet.

Die typischen Höhen von Klein- und Großbuchstaben in der Zeile werden durch Bildung von Histogramminformationen bestimmt, welche die Höhen von unzweifelhaft bestimmten Zeichen zeigen. Normalerweise zeigen derartige Histogramme zwei Spitzenniveaus, unter denen ein erstes Niveau den Kleinbuchstaben und ein zweites Niveau den Großbuchstaben entspricht.

Gewisse Zeichentypen, wie Unterstreichungen, werden zum Ende des Pufferbereichs bewegt. Dies ermöglicht es, daß diese Zeichen während des Hauptteils der Kontextanalyse effektiv ignoriert werden. Solche Zeichen werden an ihren vorhergehenden Positionen in der Zeile kurz vor Beendigung des Zeichenanalysevorgangs wiederhergestellt.

Schließlich wird eine Histogrammtypanalyse von der Breite des weißen Raums zwischen benachbarten Zeichen in der Zeile durchgeführt. Bei dem beschriebenen Beispiel werden Punktzeichen in die Histogrammanalyse nicht einbezogen, wenn die Zeile nicht vollständig aus gepunkteten Linien besteht. Typischerweise hat das Histogramm zwei Spitzenniveaus. Das erste Spitzenniveau wird angenommen als Darstellung des Zeichenabstands zwischen Einzelbuchstaben innerhalb von Worten, und das zweite Spitzenniveau wird als Wortabstandsniveau angenommen. Wenn es keine zwei unterschiedlichen Spitzenniveaus gibt, wird der Wortabstand angenommen als ein Mittelpunkt zwischen Spitzenniveaus. Wenn es nicht wenigstens zwei Spitzen gibt, wird ein willkürlicher Wortabstand auf der Basis mittlerer Höhen von Zeichen in der Zeile bestimmt.

Nach der Präparierung der Zeile für die Kontextanalyse, Block **1501**, wird ein erster Durchlauf über jedes Zeichen in der Zeile durchgeführt, Block **1502**, um Mehrdeutigkeiten zu klären. Dieser erste Durchlauf sucht nach solchen Charakteristiken, wie der relativen Höhe von Zeichen in jedem Wort, den Positionen relativ zur Grundlinie usw.. In den Fällen, bei denen die Unterscheidung zwischen einer Zahl und einem Buchstaben unklar ist, wird das Zeichen relativ zu anderen Zeichen in der Nachbarschaft analysiert, um zu bestimmen, ob dieses oder benachbarte Zeichen numerisch oder alphabetisch sind. Die Bestimmung von Zeichenmehrdeutigkeiten erfolgt nach einem iterativen Prozeß, bei dem über Nachbarzeichen bekannte Informationen für die Analyse eines speziellen Zeichens herangezogen werden. Nach der Prüfung aller Zeichen in einem Wort werden Konsistenzprüfungen durchgeführt. Wenn ein oder mehrere Zeichen mit inkonsistenten Eigenschaften ermittelt werden, werden alle Zeichen im Wort als möglicherweise falsch interpretiert ausgeflaggt. Ein zweiter Durchlauf der Kontextanalysenroutine soll die Interpretation korrigieren.

Nach Beendigung des ersten Kontextanalysendurchlaufs für jede Zeile in der Fahne versucht die Kontextanalysenroutine, eine Schriftart-Identifizierungsnummer jedem Zeichen in der Fahne zuzuordnen und die Zeichengrößen für jede Schriftart zu bestimmen, Block **1503**. Die Schriftartidentifizierung wird über die Fahne fortgesetzt, indem alle Zeichen, die durch Schablonen identifiziert wurden, verfolgt werden. Alle diejenigen Zeichen, welche mit einer speziellen Schablone übereinstimmen, werden miteinander in einer Verknüpfungsliste verknüpft, wobei die Verknüpfungsliste eine auf eine besondere Schablone hinweisende Wurzel hat. Auf der Basis dieser verknüpften Zeichenlisten werden unter der Bedingung Worte zu Schriftarten zugeordnet, daß Worte mit von derselben Schablone identifizierten Zeichen dieselbe Schriftart haben. Dieser ist ein langer iterativer Prozeß. Nachher werden Histogramminformationen erstellt, welche die Höhe von Groß- und Kleinbuchstaben in jeder Schriftart detailliert beschreiben.

Danach wird ein zweiter Durchlauf über jede Zeile in der Fahne durchgeführt, Block **1504**. Während des ersten Durchlaufs als unstimmig ausgeflaggte Worte werden erneut analysiert, um festzustellen, welche Buchstaben unrichtig sind. Der zweite Durchlauf prüft solche Kennzeichen, wie Grundliniengleichmäßigkeit, Zeichengrößengleichmäßigkeit, alphabetischen/numerischen Kontext usw..

Schließlich wird ein Reihe von gemischten Säuberungsroutinen verwendet, Block **1505**. Hierzu gehören die Prüfung der Zeichensetzung zur Gewährleistung einer vernünftigen Größe und Position. Beispielsweise wird ein übergroßes Satzzeichen wahrscheinlich als unerkanntes Zeichen behandelt und als solches ausgeflaggt. Punkte und relativ breite

Kommas oder Apostrophe, die über der Basislinie und unterhalb des oberen Endes des Kleinbuchstabens liegen, können kurze Gedankenstriche oder Punktfolgen sein. Diese Zeichen werden dem ASCII-Code für einen Gedanken- oder Bindestrich zugeordnet. Bei dem Versuch, in einem Bit-mapped-Bild, das einen nicht-gedruckten Text darstellt, etwas wie eine Unterschrift oder ein Logo zu erkennen, ist das Ergebnis typischerweise eine Folge mit unerkannten Zeichen und ein Satz anderer Zeichen, wie Satzzeichen und Gedankenstrichen. Derartige Folgen kollabieren in einem einzigen unerkannten Zeichen. Aufeinanderfolgende einzelne Angaben werden in doppelte Anführungszeichen zusammengefaßt.

Die Kontextanalysenroutine versucht auch, durch die Zeichenerkennungsroutinen getrennte Zeichen zu verschmelzen. So können beispielsweise zwei Zeichen, die von den Zeichenerkennungsroutinen als offene Klammern erkannt worden sind "(", gefolgt von einer geschlossenen Klammer")" tatsächlich den Buchstaben "O" darstellen. Die Kontextanalysenroutinen versuchen, derartig gesplittete Zeichen dadurch zu vereinigen, daß sie die Positionsnähe und die Zusammengehörigkeit spezieller Zeichenpaare erkennt.

Andere Zeichen, wie Unterstreichungen, werden analysiert, um festzustellen, ob sie innerhalb einer vorgegebenen Distanz von der Basislinie liegen. Wenn die Unterstreichung mehr als eine vorgegebene Distanz entfernt ist oder wenn ihre Ränder nicht mit den Wortgrenzen zusammenfallen, wird sie als Linie und nicht als Unterstreichung aufgefaßt. Andererseits werden die Zeichen oberhalb der Unterstreichung als unterstrichen ausgeflaggt. Die vorgegebene Distanz wird auf der Basis des relativen Kontextes der Zeile, einschließlich solcher Faktoren wie Größe der Buchstaben, bestimmt.

Die Kontextanalysenroutinen versuchen, nicht-identifizierte Zeichen durch Verschmelzen unterbrochener Zeichenteile, durch Wiedereinführen von Zeichen in die Zeichenerkennungsroutinen und die Anlegung weniger scharfer Beschränkungen für die Erkennung zu identifizieren.

Am Ausgang der Kontextanalysenroutinen erscheint die vollständige abgetastete Seite mit ASCII-Zeichendarstellungen für die Zeichen auf der Seite in der normalen Lesefolge der Zeichen.

Patentansprüche

1. Verfahren zum Erkennen von Zeichen auf einem Medium
wobei das Medium abgetastet und ein Bit-mapped-Bild des Mediums erzeugt wird;
wobei das Bit-mapped-Bild einer Strukturanalyse **(103, 203, 232)** zum Isolieren einzelner Zeichen unterworfen wird und als Ausgabe der Strukturanalyse ein Bit-mapped-Bild der Zeichen erzeugt wird;
wobei die Zeichen vorläufig identifiziert werden **(234)**; und
wobei jedes Zeichen auf der Basis des das Zeichen im Medium umgebenden Kontextes analysiert wird **(104)**;
dadurch gekennzeichnet,
a) daß eine Textzeile in dem Medium zum Bestimmen von räumlichen Informationen über die Textzeile analysiert wird;
b) daß Attributdaten für jedes vorläufig identifizierte Zeichen in der Textzeile gewonnen und diesem Zeichen zugeordnet werden; und
c) daß Mehrdeutigkeiten bei einem vorläufig identifizierten Zeichen mit Hilfe der räumlichen Informationen über die Textzeile und mit Hilfe der zugeordneten Attributdaten aufgelöst werden **(1502)**.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß die räumlichen Informationen Informationen über den Schrägfehler der Textzeile, den Zeichenabstand und über Zeichenhöhen **(1503)** umfassen.

3. Verfahren nach Anspruch 1 oder 2, dadurch gekennzeichnet,
daß ein Histogramm der Abstände zwischen den Zeichen in der Textzeile erzeugt wird; und
daß die mittlere Höhe von unzweifelhaft identifizierten Zeichen in der Textzeile bestimmt wird **(1503)**.

4. Verfahren nach einem der Ansprüche 1 bis 3, dadurch gekennzeichnet,
daß eine Datenbank von charakteristischen Attributen bekannter Zeichen vorgesehen wird; und
daß auf die Datenbank zum Gewinnen von charakteristischen Attributen des mehrdeutigen Zeichens zugegriffen wird.

5. Anordnung zum Erkennen von Zeichen auf einem Medium mit
einer Scannereinrichtung, die ausgangsseitig ein Bit-mapped-Bild des Mediums erzeugt;
einem mit der Scannereinrichtung gekoppelten Speicher zum Speichern des Bit-mapped-Bildes;
einer mit dem Speicher gekoppelten Prozessoreinrichtung zur Strukturanalyse des Bit-mapped-Bildes des Mediums und zur Ausgabe einer Bit-mapped-Darstellung der Zeichen;
einer Einrichtung zum vorläufigen Identifizieren der Zeichen und
einer Einrichtung zum Analysieren jedes Zeichens auf der Basis des das Zeichen im Medium umgebenden Kontextes, wobei die Einrichtung zum Analysieren gekennzeichnet ist durch:
eine erste Einrichtung zum Analysieren einer Textzeile zum Bestimmen von räumlichen Informationen über die Textzeile im Medium;
eine zweite Einrichtung zum Gewinnen und Zuordnen von Attributdaten für jedes vorläufig identifizierte Zeichen in der Textzeile; und
eine mit der ersten und der zweiten Einrichtung gekoppelte Einrichtung zum Auflösen von Mehrdeutigkeiten bei einem vorläufig identifizierten Zeichen mit Hilfe der räumlichen Informationen über die Textzeile und der zugeordneten Attributdaten.

6. Anordnung nach Anspruch 5, dadurch gekennzeichnet, daß die Einrichtung zum Analysieren einer Textzeile eine Einrichtung zum Beschreiben des Schrägfehlers der Textzeile, von Zeichenabstandsinformationen und von Zeichenhöhen aufweist.

7. Anordnung nach Anspruch 5 oder 6, wobei die Einrichtung zum Analysieren einer Textzeile eine Einrichtung zum Erzeugen eines Histogramms der Abstände zwischen den Zeichen in der Textzeile und eine Einrichtung zum Bestimmen der mittleren Höhe von unzweifelhaft identifizierten Zeichen in der Textzeile umfasst.

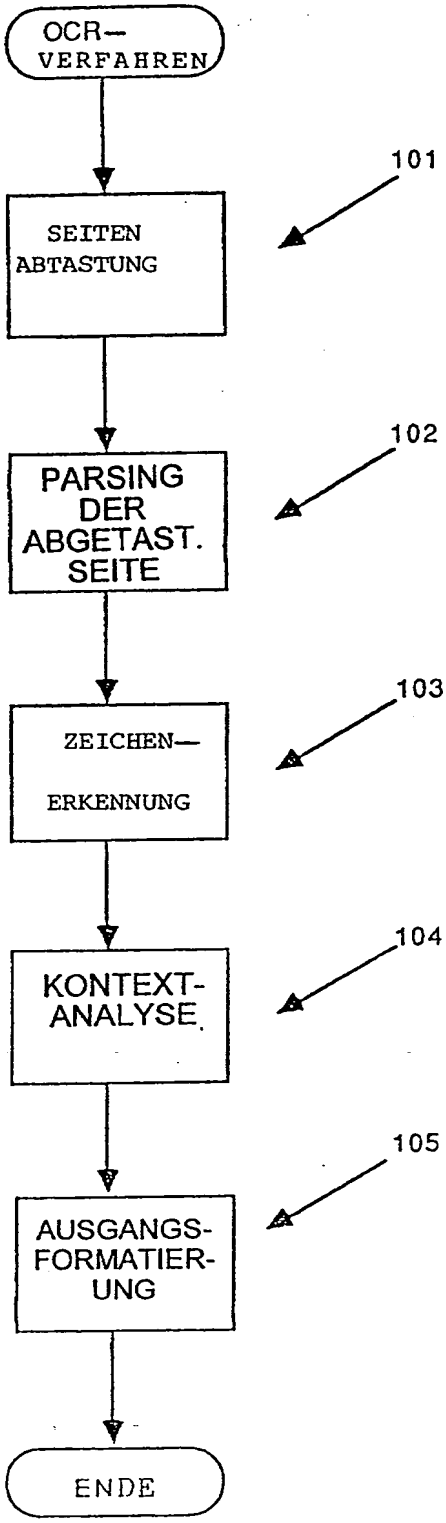
8. Anordnung nach einem der Ansprüche 5 bis 7, gekennzeichnet durch

eine Datenbank-Einrichtung für charakteristische Attribute von bekannten Zeichen; und durch
eine Einrichtung zum Zugreifen auf die Datenbank-Einrichtung zum Gewinnen von charakteristischen Attributen eines mehrdeutigen Zeichens.

Hierzu 21 Seite(n) Zeichnungen

- Leerseite -

Nummer: DE 39 26 327 C2
In G 06 K 9/60
Veröffentlichungstag: 18. März 1999



FIGUR 1

Nummer: DE 39 26 327 C2
Int. Cl.[6]: G 06 K 9/60
Veröffentlichungstag: 18. März 1999

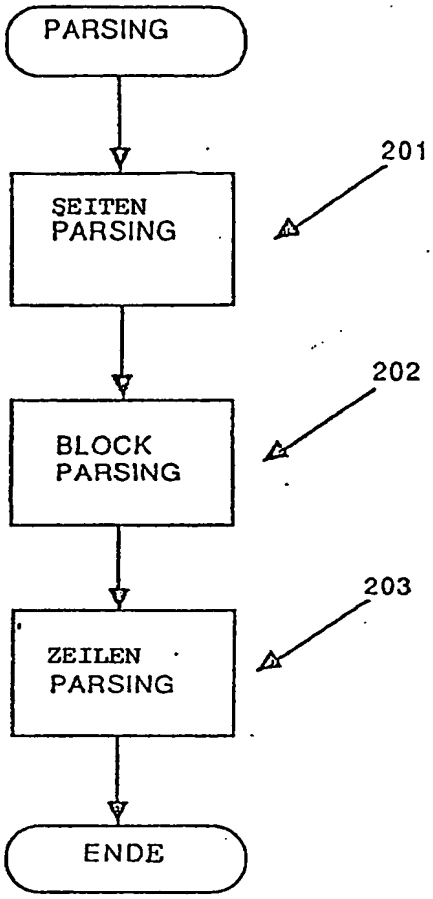
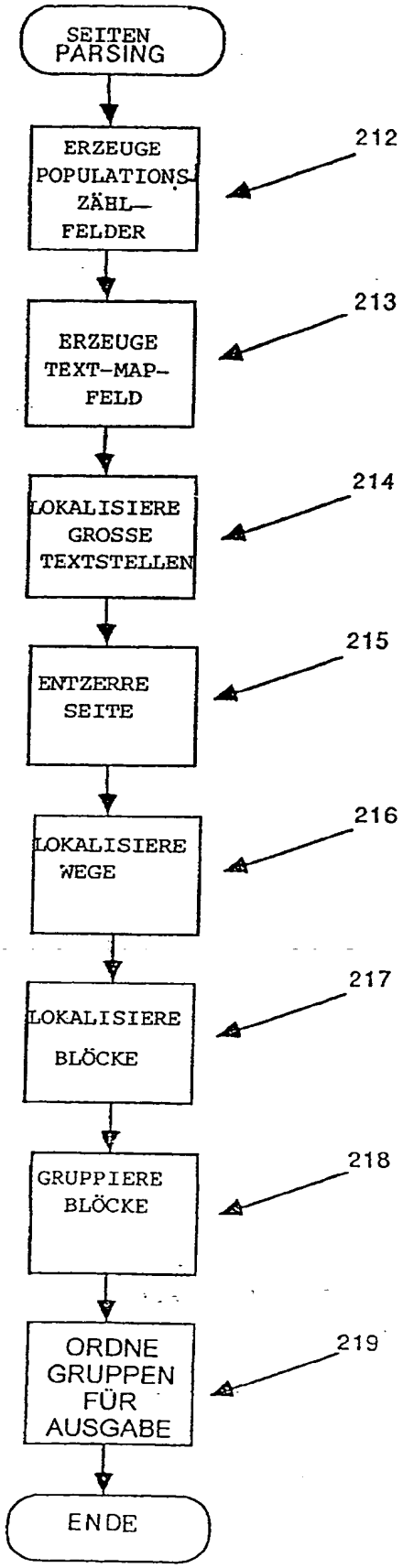


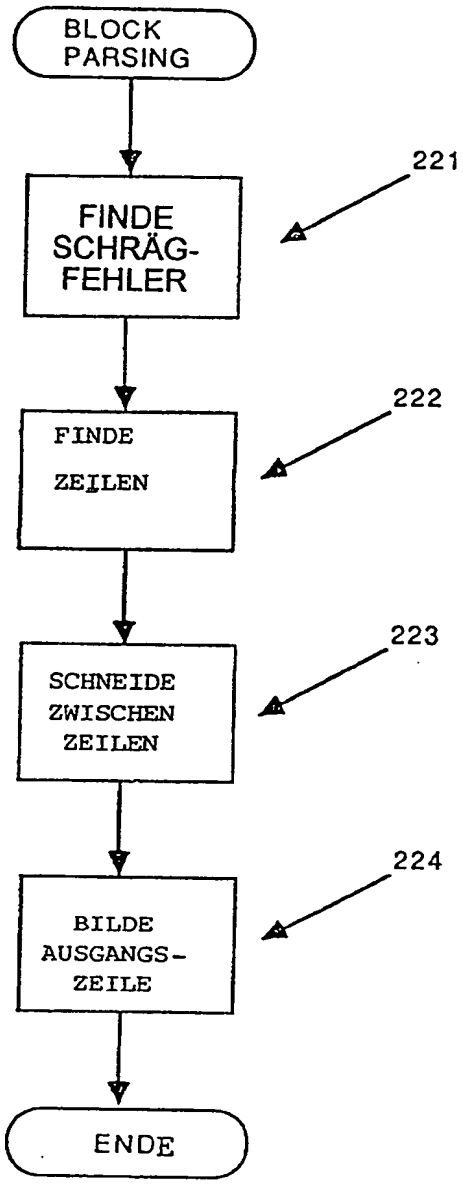
FIGURE 2 (a)

Nummer: DE 39 26 327 C2
In: G 06 K 9/60
Veröffentlichungstag: 18. März 1999



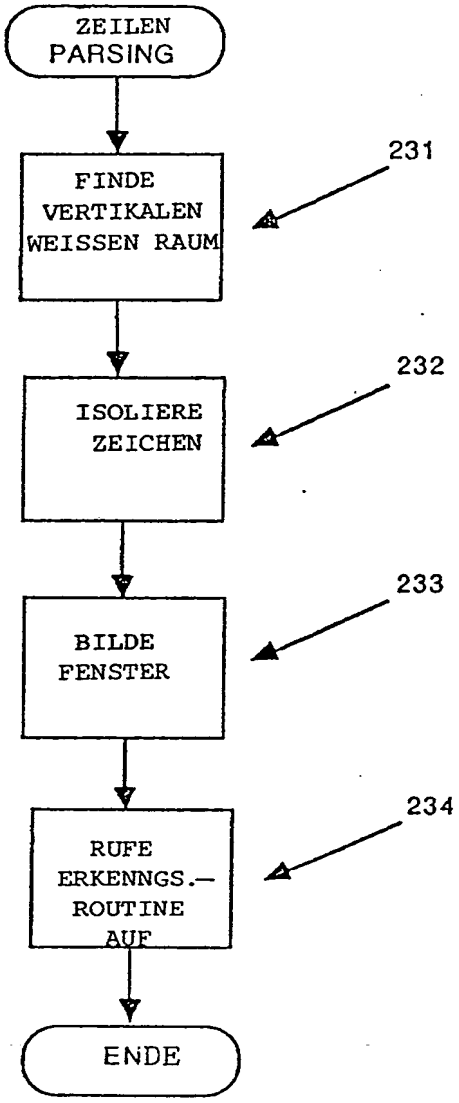
FIGUR 2 (b)

Nummer: DE 39 26 327 C2
Int. Cl.[6]: G 06 K 9/60
Veröffentlichungstag: 18. März 1999



FIGUR 2 (c)

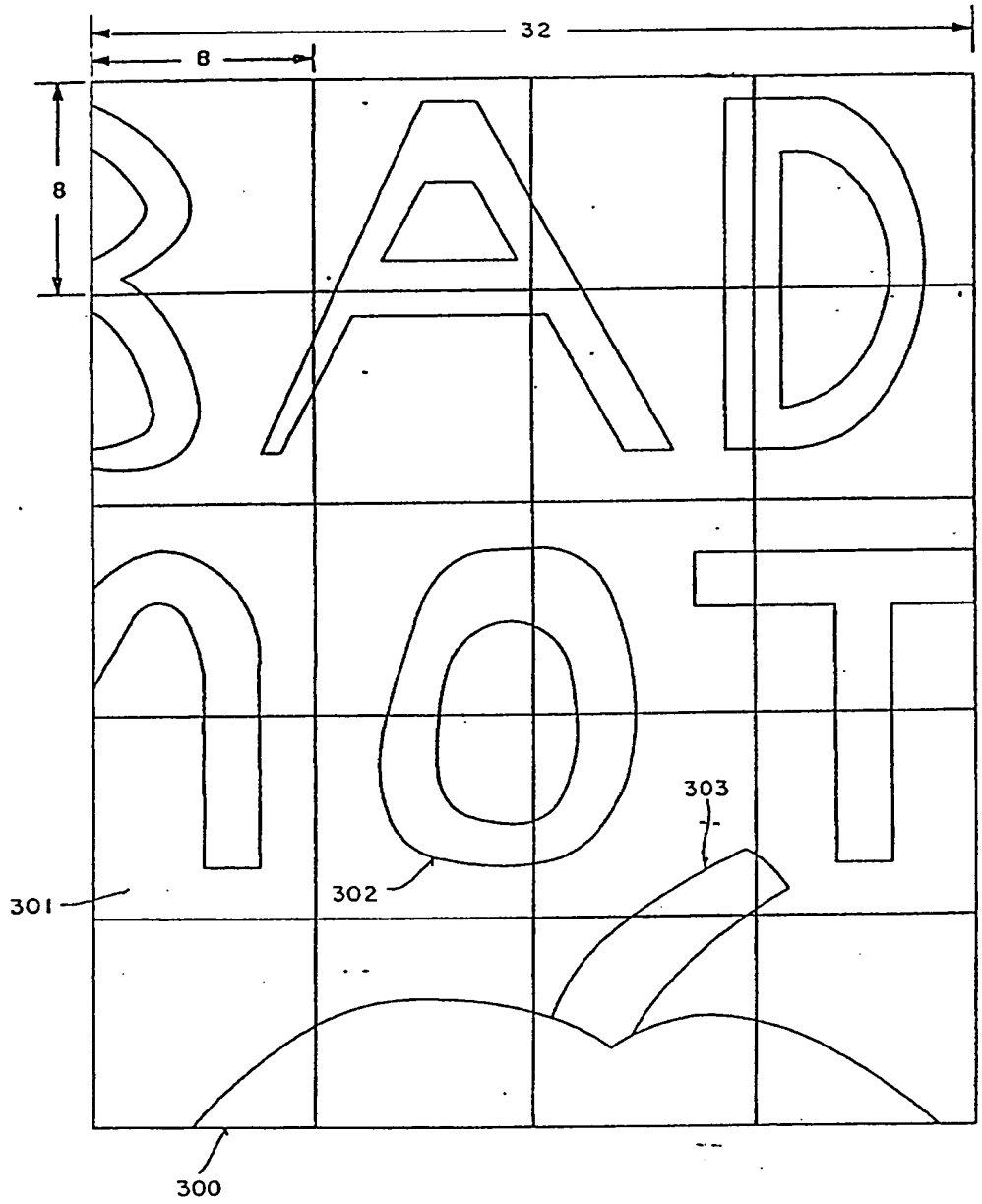
Nummer: DE 39 26 327 C2
In[illegible] G 06 K 9/60
Veröffentlichungstag: 18. März 1999



FIGUR 2 (d)

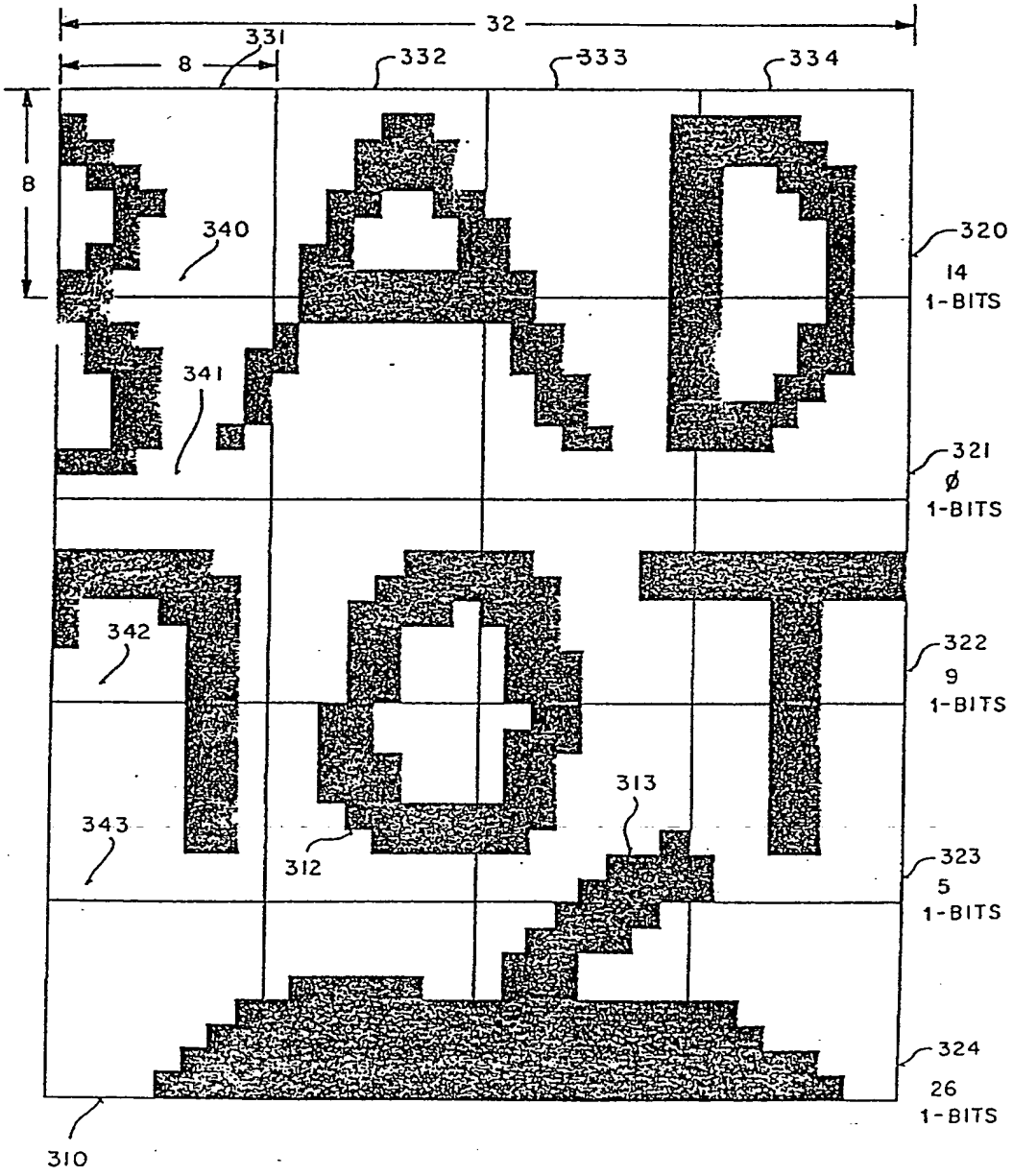
Nummer: DE 39 26 327 C2
Int. Cl.[6]: G 06 K 9/60
Veröffentlichungstag: 18. März 1999

FIG 3(a)

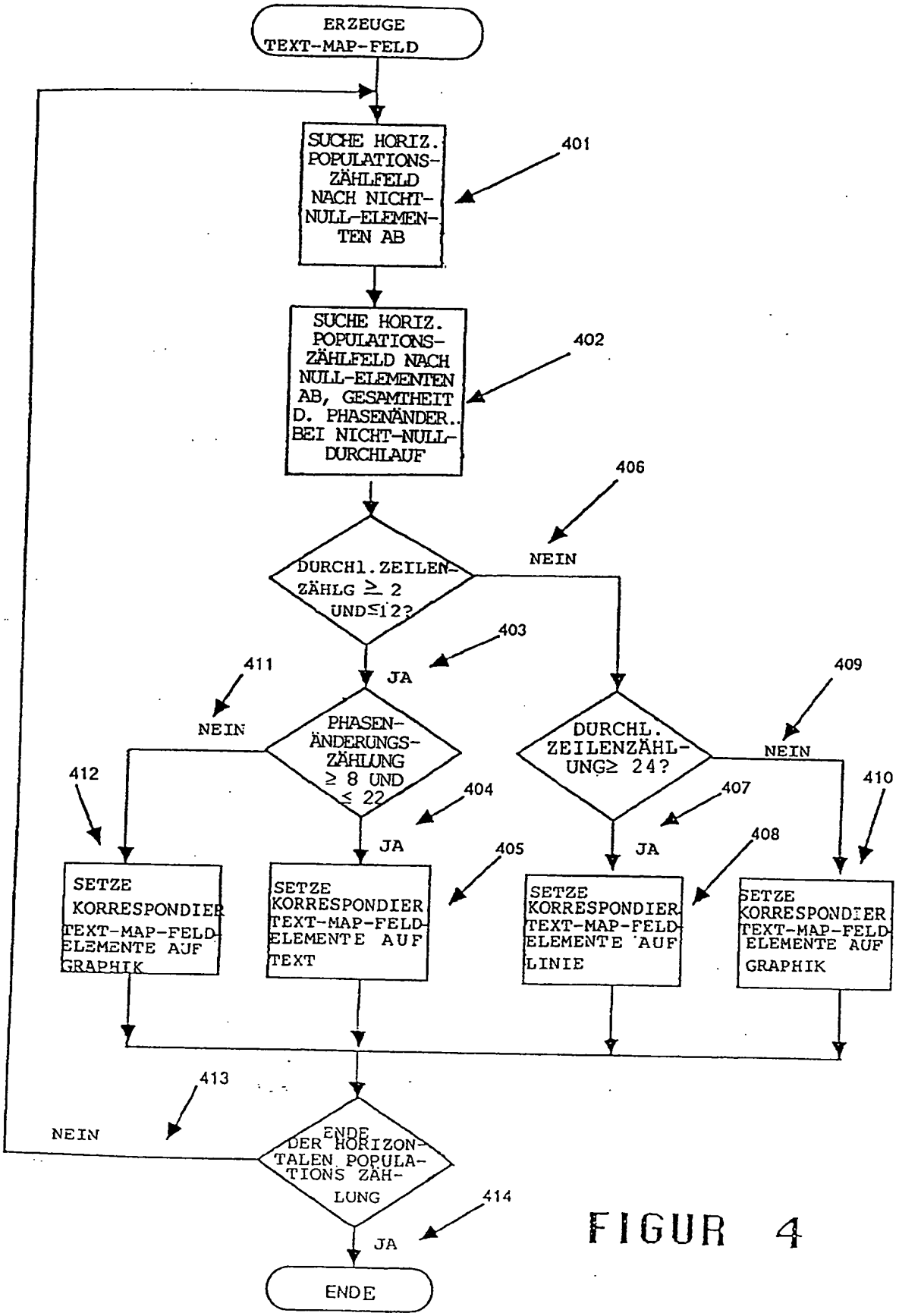


Nummer: DE 39 26 327 C2
In[redacted] G 06 K 9/60
Veröffentlichungstag: 18. März 1999

FIG 3(b)

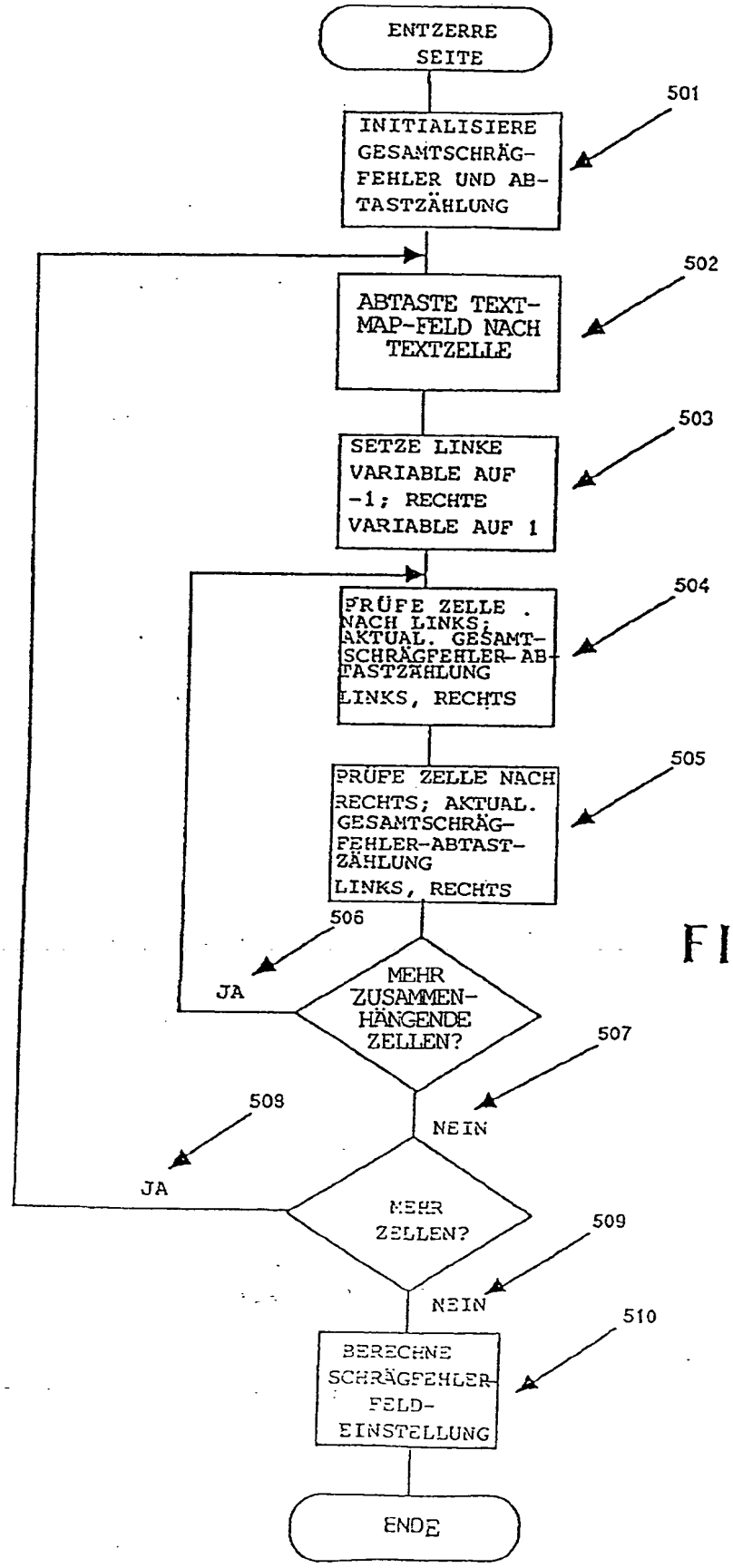


Nummer: DE 39 26 327 C2
Int. Cl.[6]: G 06 K 9/60
Veröffentlichungstag: 18. März 1999



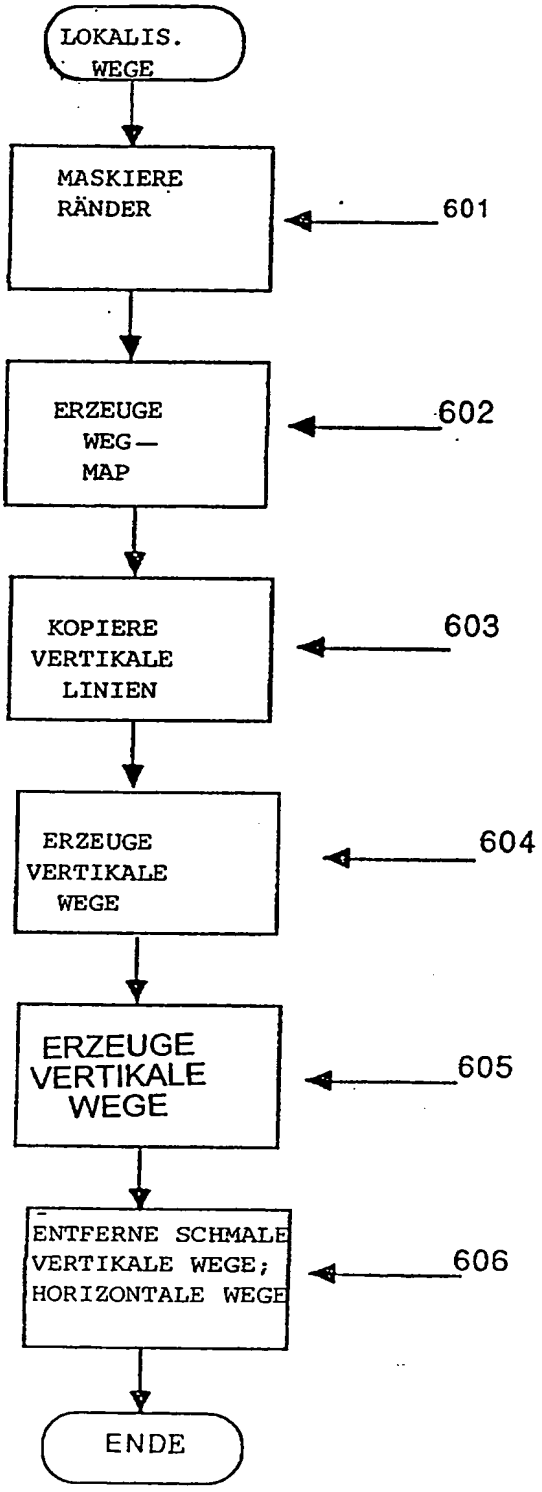
FIGUR 4

Nummer: DE 39 26 327 C2
In[redacted] G 06 K 9/60
Veröffentlichungstag: 18. März 1999



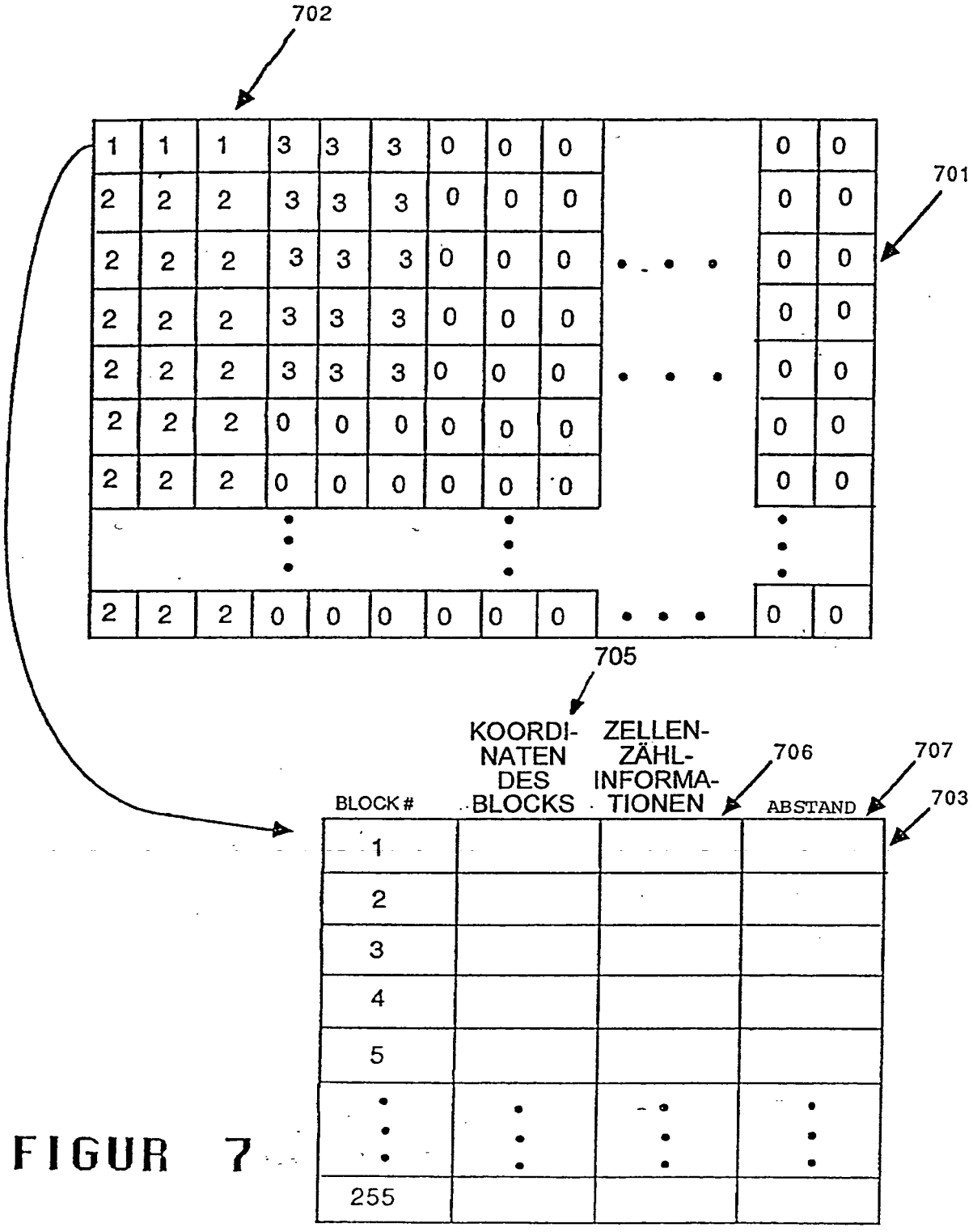
FIGUR 5

Nummer: DE 39 26 327 C2
Int. Cl.[6]: G 06 K 9/60
Veröffentlichungstag: 18. März 1999



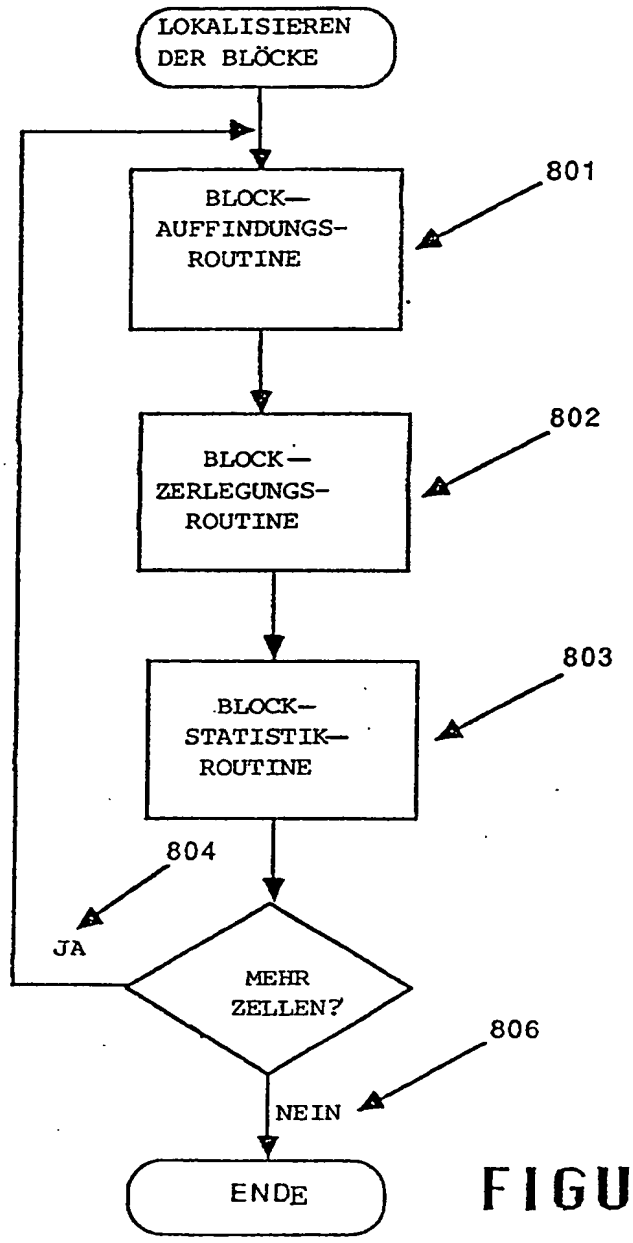
FIGUR 6

Nummer: DE 39 26 327 C2
In[redacted]: G 06 K 9/60
Veröffentlichungstag: 18. März 1999



FIGUR 7

Nummer: DE 39 26 327 C2
Int. Cl.[6]: G 06 K 9/60
Veröffentlichungstag: 18. März 1999



FIGUR 8

Nummer: DE 39 26 327 C2
Int. Cl.: G 06 K 9/60
Veröffentlichungstag: 18. März 1999

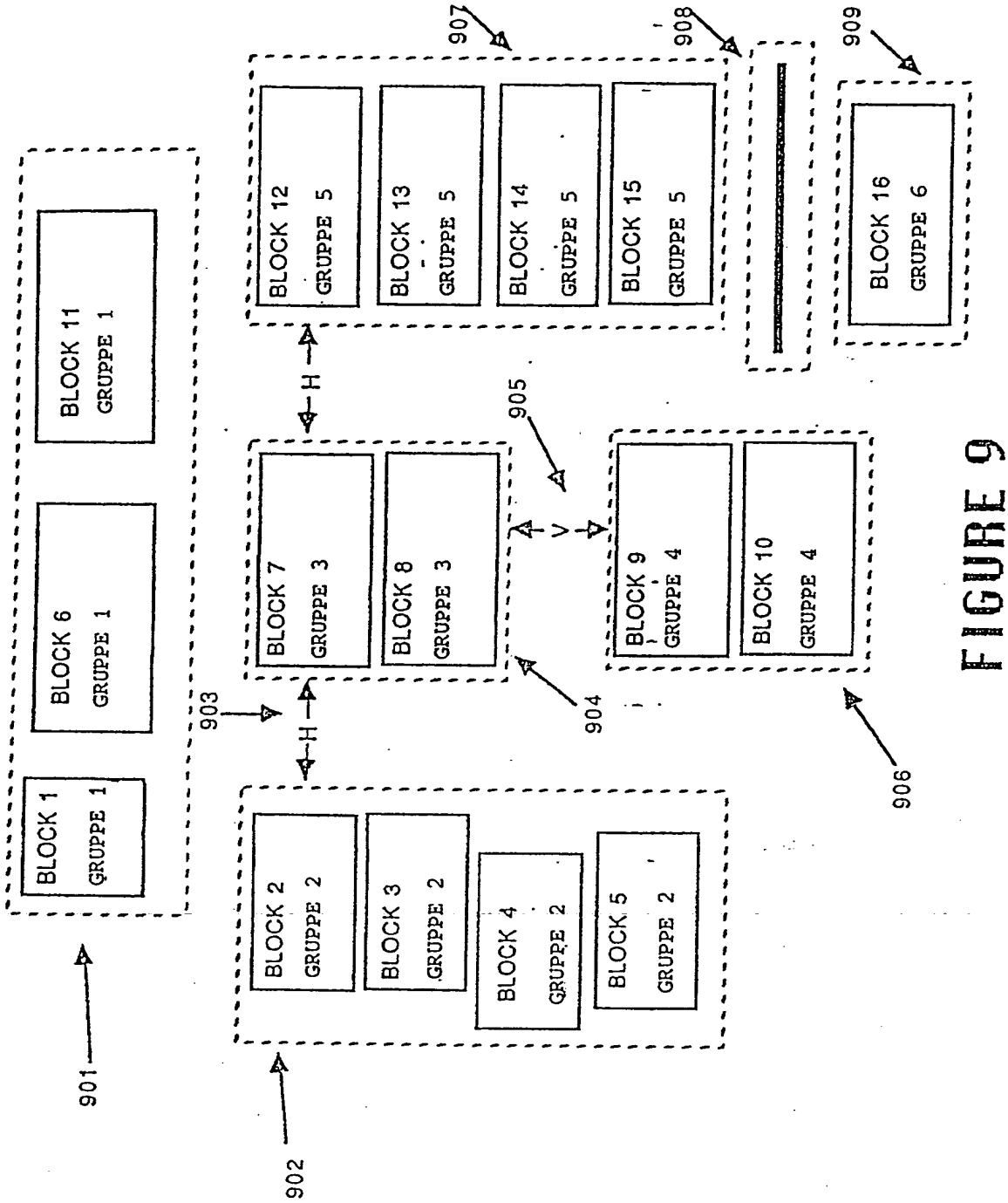


FIGURE 9

Nummer: DE 39 26 327 C2
Int. Cl.[6]: G 06 K 9/60
Veröffentlichungstag: 18. März 1999

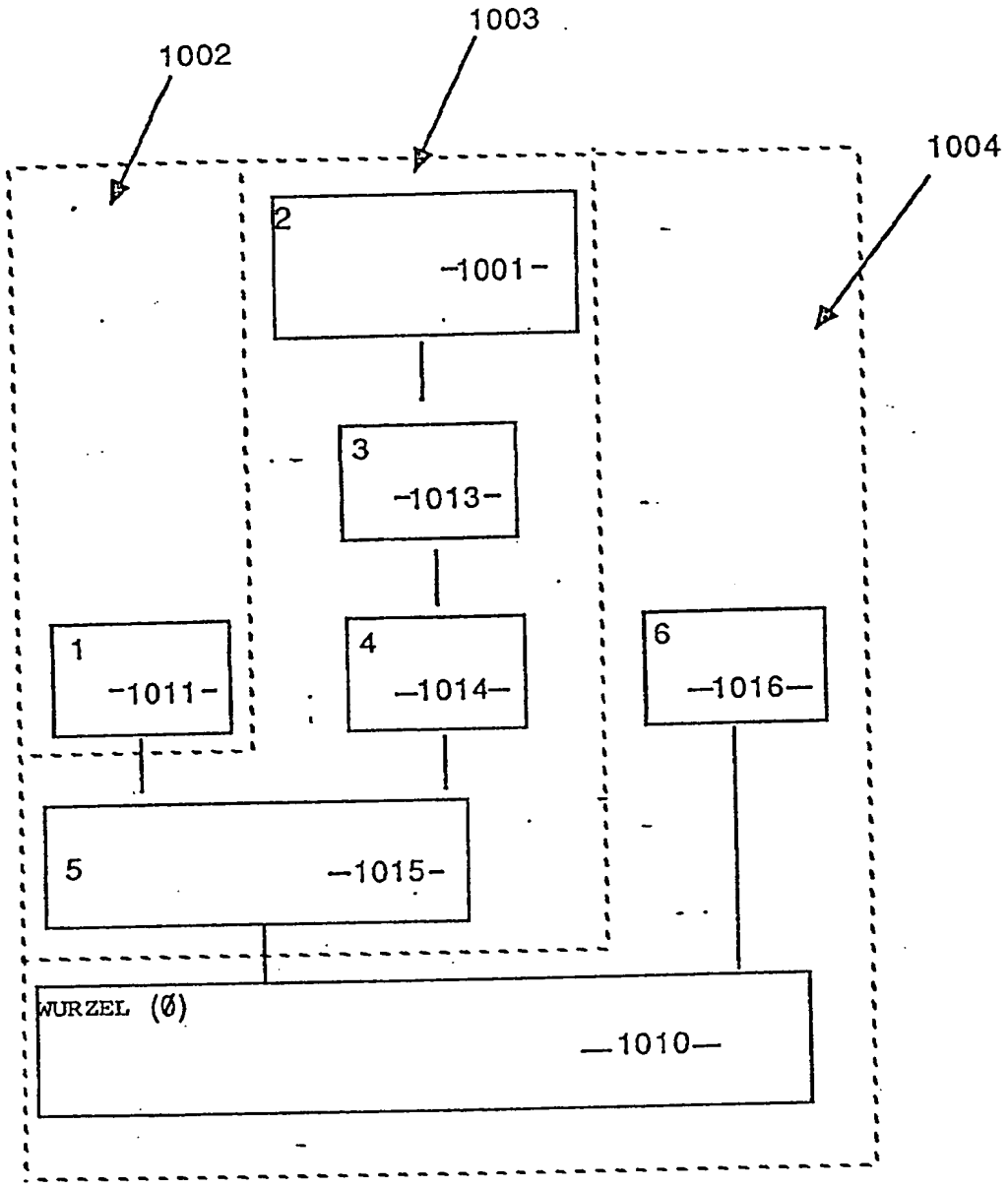


FIGURE 10 (a)

Nummer: DE 39 26 327 C2
Int. Cl.: G 06 K 9/60
Veröffentlichungstag: 18. März 1999

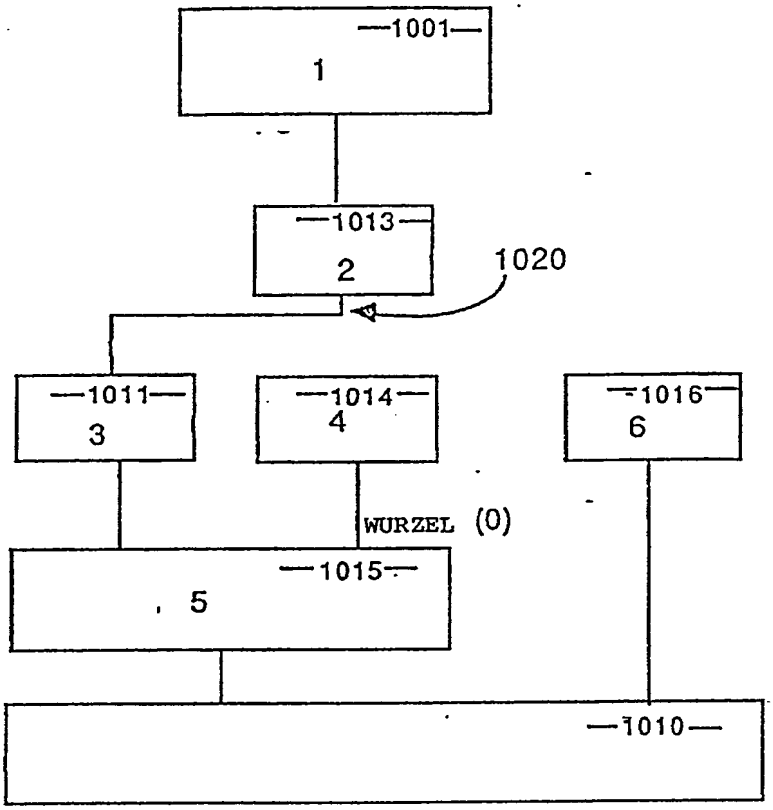
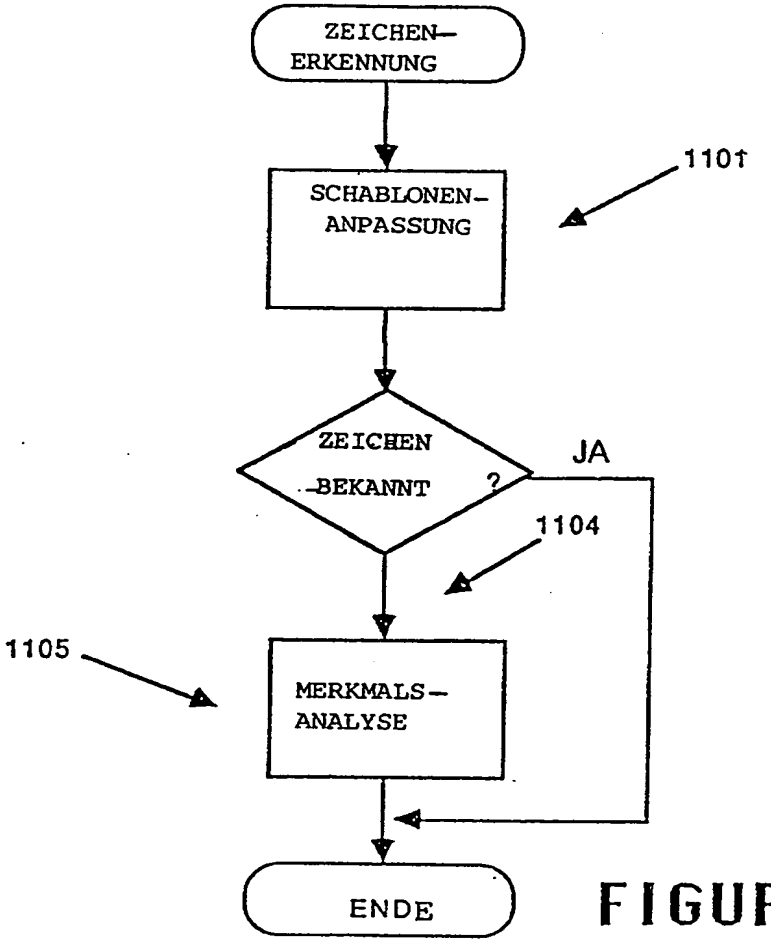


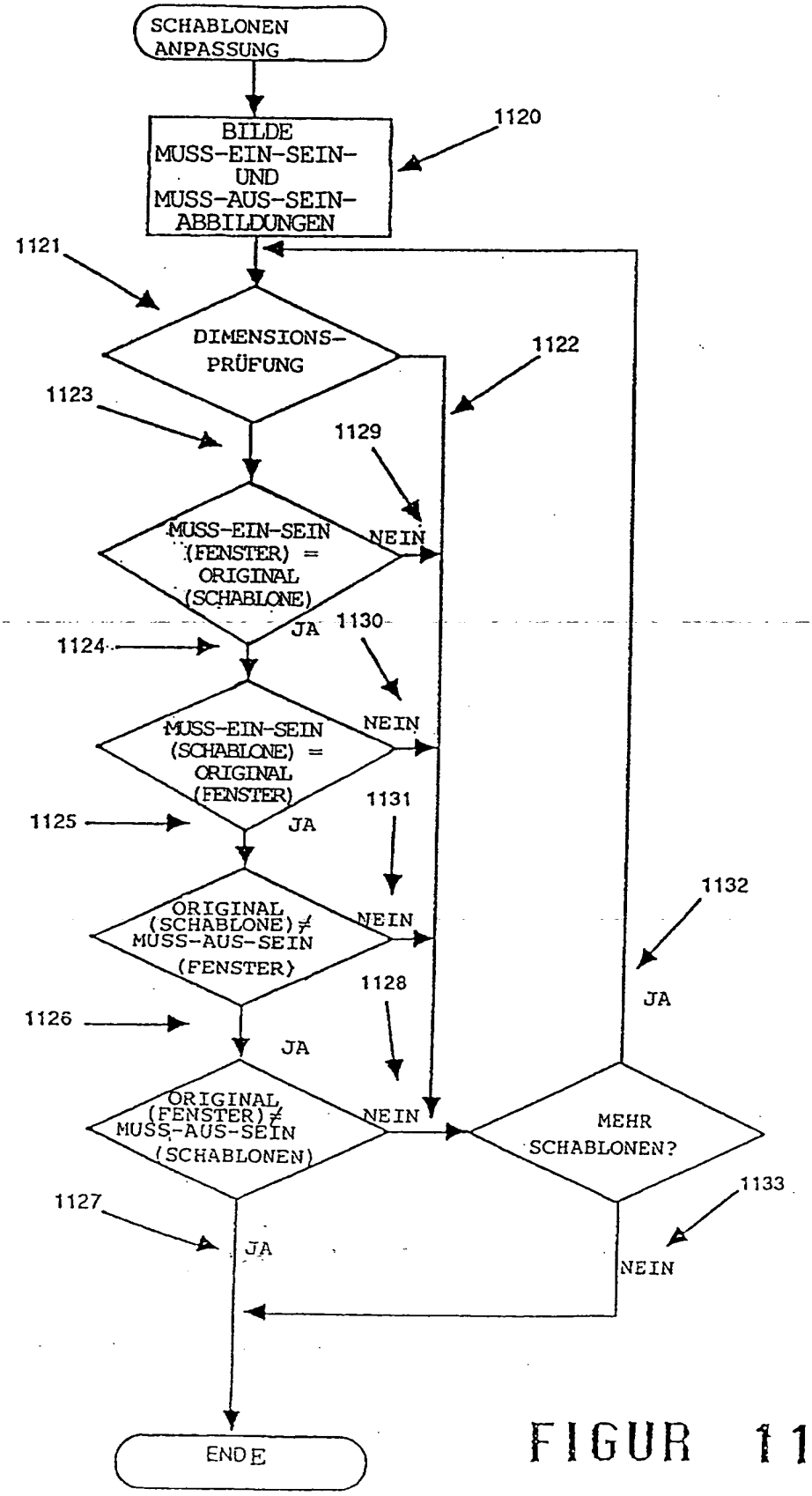
FIGURE 10 (b)

Nummer: DE 39 26 327 C2
Int. Cl.[6]: G 06 K 9/60
Veröffentlichungstag: 18. März 1999



FIGUR 11 (a)

Nummer: DE 39 26 327 C2
Int. Cl.: G 06 K 9/60
Veröffentlichungstag: 18. März 1999



FIGUR 11 (b)

Nummer: DE 39 26 327 C2
Int. Cl.[6]: G 06 K 9/60
Veröffentlichungstag: 18. März 1999

FIG 12(a)

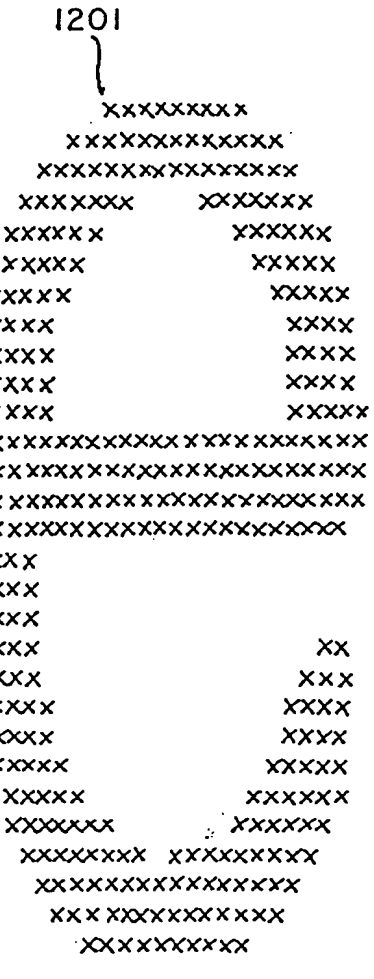
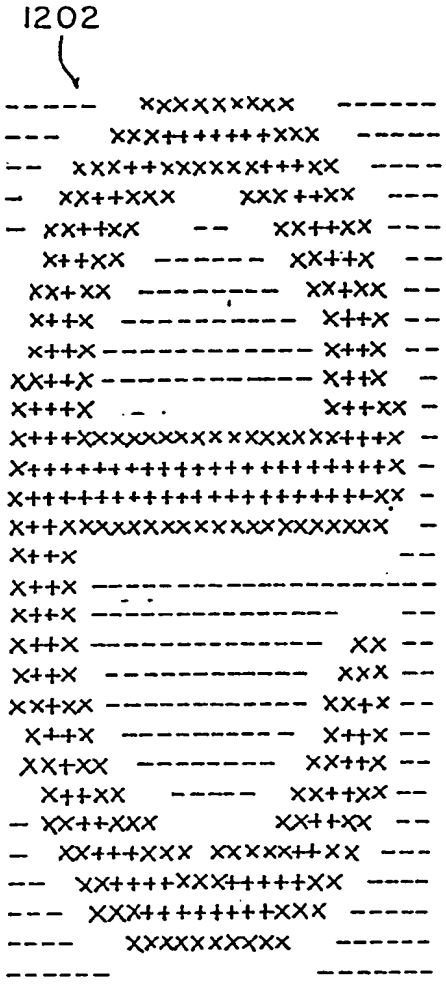
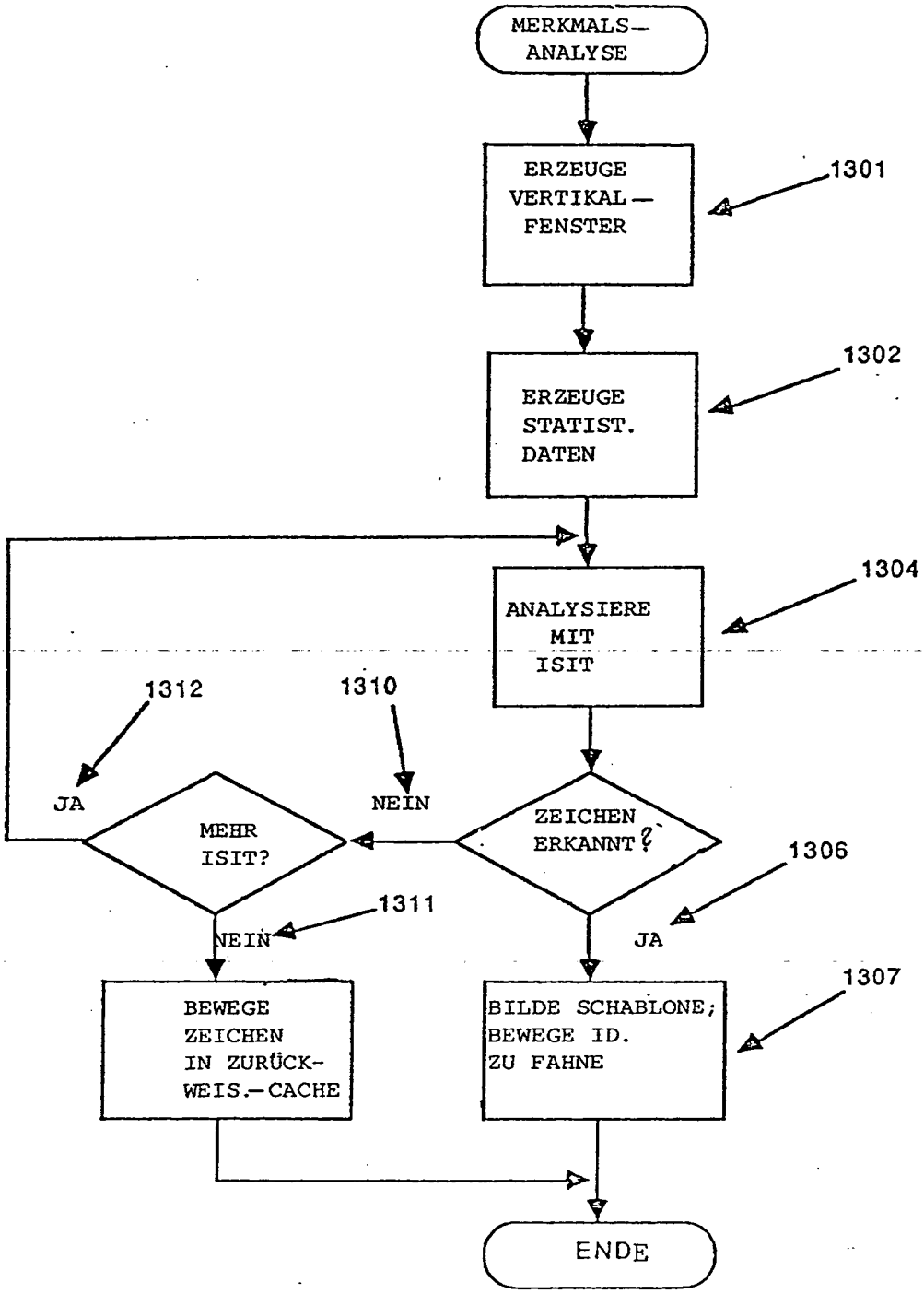


FIG 12(b)



Nummer: DE 39 26 327 C2
Int. Cl.: G 06 K 9/60
Veröffentlichungstag: 18. März 1999



FIGUR 13

Nummer: DE 39 26 327 C2
Int. Cl.[6]: G 06 K 9/60
Veröffentlichungstag: 18. März 1999

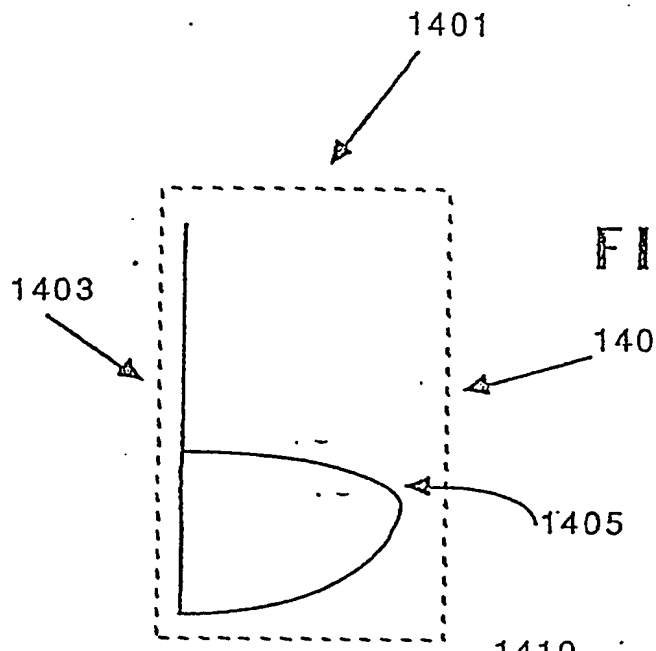


FIGURE 14 (a)

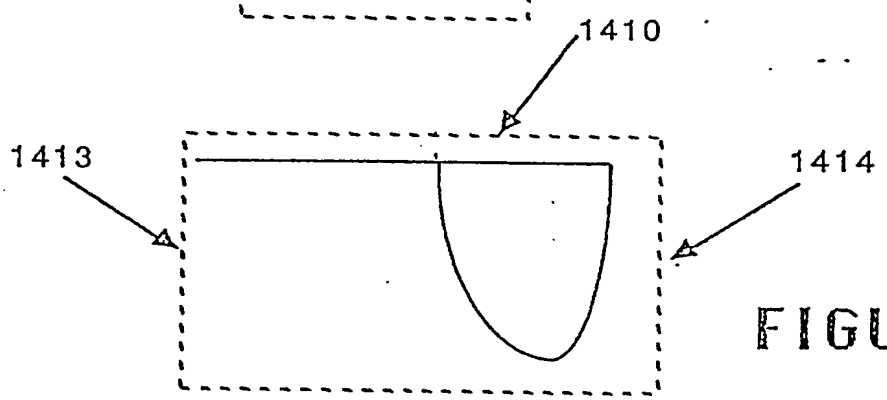


FIGURE 14 (b)

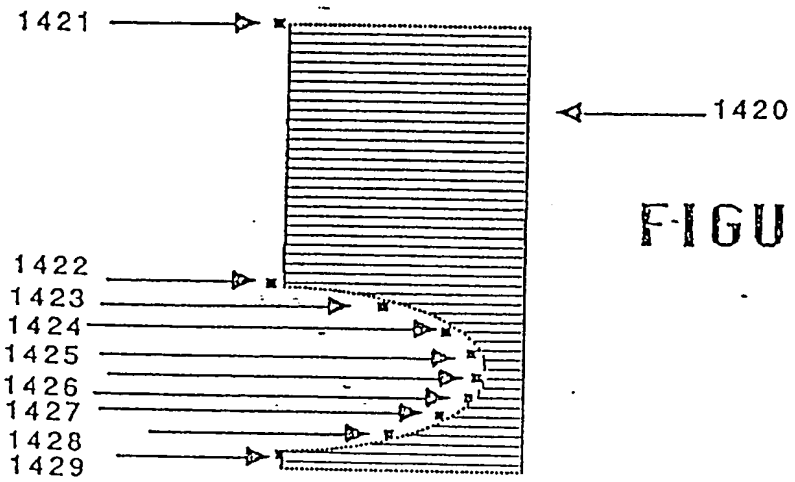
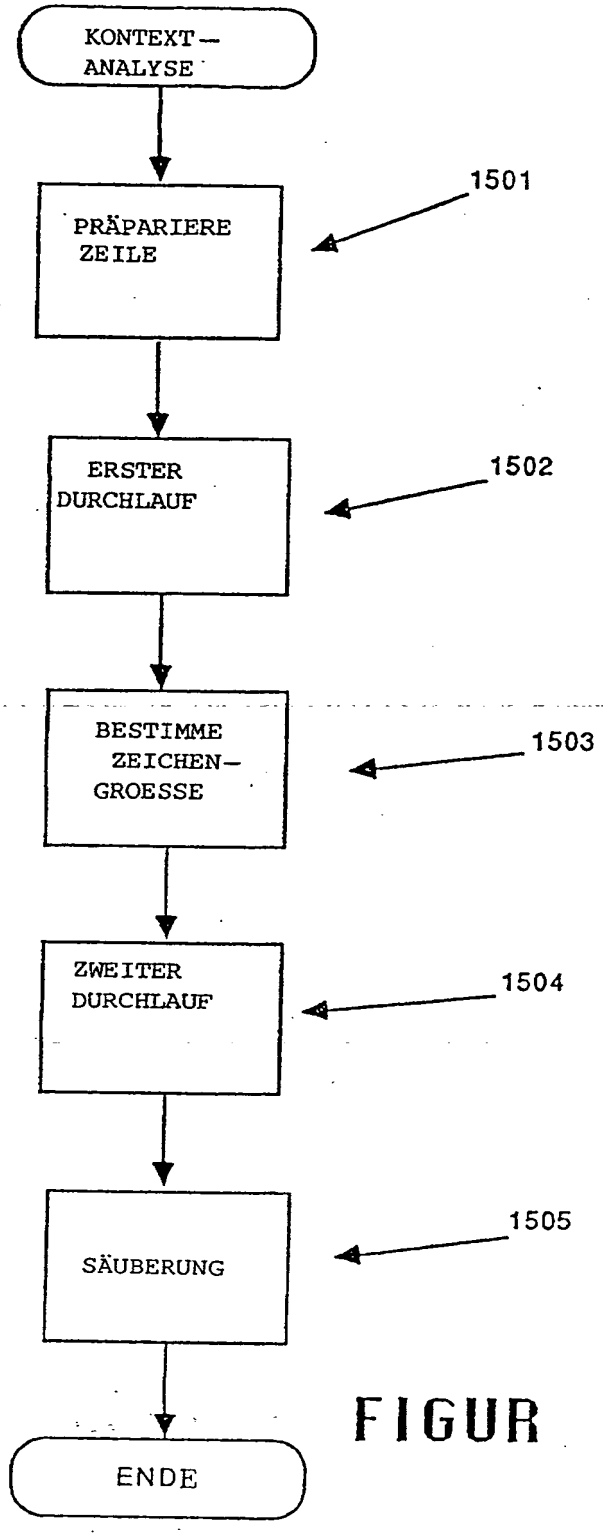


FIGURE 14 (c)

Nummer: DE 39 26 327 C2
Int. Cl.: G 06 K 9/60
Veröffentlichungstag: 18. März 1999



FIGUR 15